Bauhaus-Universität Weimar
Fakultät Medien
Studiengang Medieninformatik

Worteinbettung als semantisches Feature in der argumentativen Analyse

Bachelorarbeit

Kevin Lang
geb. am: 05.09.1989 in Greiz

Matrikelnummer 110010

1. Gutachter: Prof. Dr. Benno Stein
2. Gutachter: Prof. Dr.-Ing. Volker Rodehorst

Datum der Abgabe: 8. Februar 2016

Erklärung

Hiermit versichere ich, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Weimar, 8. Februar 2016

.....
Kevin Lang

Zusammenfassung

Diese Arbeit beschäftigt sich mit der Nutzung von Worteinbettungen in der automatischen Analyse von argumentativen Texten. Die Arbeit diskutiert wichtige Einstellungen des Einbettungsverfahren sowie diverse Anwendungsmethoden der eingebetteten Wortvektoren für drei Aufgaben der automatischen argumentativen Analyse: Textsegmentierung, Argumentativitäts-Klassifikation und Relationenfindung. Meine Experimente auf zwei Standard-Argumentationsdatensätzen zeigen die folgenden Haupterkenntnisse: Bei der Textsegmentierung konnten keine Verbesserungen erzielt werden, während in der Argumentativitäts-Klassifikation und der Relationenfindung sich kleine Erfolge gezeigt haben und weitere bestimmte Forschungsthesen bewahrheitet werden konnten. In der Diskussion wird darauf eingegangen, warum bei der einfachen Worteinbettung in der argumentativen Analyse sich kaum nutzbare Ergebnisse erzielen lassen konnten, diese sich aber in Zukunft durch erweiterte Worteinbettungsverfahren verbessern können.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die Forschung an der automatischen Analyse argumentativer Texte hat in den letzten Jahren sehr an Bedeutung gewonnen. Durch die weite Verbreitung von Internet-Foren und sozialen Netzwerken werden jeden Tag unzählig viele Diskussionen und Beiträge verfasst. Die Interessenten an diesen Themen möchten aber nicht immer den gesamten Text lesen müssen, sondern aus Zeitgründen nur seine argumentativen Kernaussagen verstehen. Um diese Aussagen schneller zu erkennen und aus ihnen zu lernen, haben es sich viele Forschungsgruppen zur Aufgabe gemacht solche Texte zu analysieren.

Bereits seit den 80er Jahren gibt es verschiedene Modelle um rhetorische Strukturen in Texten darzustellen, aber besonders bezüglich argumentativer Texteinheiten wurde bis heute kein einheitlicher Standard festgelegt. Ein Text gilt als argumentativ, wenn eine Person seine Leser durch Zuhilfenahme von Aussagen von einer oder mehrerer Behauptungen überzeugen will. Diese Aussagen können die Behauptungen unterstützen, gegebenenfalls aber auch attackieren. Der zweite Fall tritt seltener auf, wird aber dennoch in Argumentationen verwendet, um Vor- und Nachteile eines Themas zu verdeutlichen. In der argumentativen Analyse werden die Behauptungen meist als 'Claim' bezeichnet und die dazugehörigen Aussagen als 'Premise' (Stab and Gurevych [2014]).

Die automatische argumentative Analyse von Texten soll es ermöglichen Kernaussagen einer Argumentationskette zu identifizieren, die Struktur einer Argumentationskette zu überprüfen und neue Argumente zu gegebenen Themen innerhalb einer Dokumentsammlung zu finden. Dies ermöglicht bisherige Argumentationen besser zu verstehen und in Zukunft logischer aufzubauen. Ein weiterer Anwendungsfall ist zum Beispiel gleiche Argumentationen in verschiedenen Dokumenten oder gar in einer Rede zu finden, um Redundanz zu erkennen. So lassen sich neue Argumentationen von Wiederholungen trennen,

damit man sehen kann, ob wirklich neue Aspekte in eine Diskussion eingebracht wurden.

In meiner Arbeit verfolge ich den neuen Ansatz Probleme in der automatischen argumentativen Analyse von englischen Texten zu lösen, indem ich versuche die semantischen Relationen zwischen Wortpaaren, Segmenten oder ganzen Sätzen zu erkennen. In bisherigen Arbeiten wurde der semantische Aspekt oft vernachlässigt oder nur anhand von bestimmten Wortindikatoren beurteilt. Ich versuche durch das Lernen von Worteinbettungen mit Hilfe des Frameworks 'Word2Vec' (Mikolov et al. [2013]) die semantischen Relationen zwischen Wörtern zu erkennen, wie zum Beispiel zwischen 'Bibliothek' und 'Buch' oder 'Auto' und 'Reifen'. Mit den dadurch aufgestellten Wortvektoren können Beziehungen zwischen Wörtern erkannt werden, welche ich für die Erkennung von Relationen zwischen argumentativen Einheiten benutze. Ähnlich wie beim Lernprozess des Menschen ist es dabei wichtig für jede Aufgabe die richtigen Trainingsdaten für die Worteinbettungen zu wählen. Dies ist vergleichbar mit dem Lernen von Vokabeln für ein Sprachexamen oder das Lernen von Ländern und Städten für ein Geografieexamen. Deswegen habe ich Trainingskorpora gewählt, die einerseits argumentative Inhalte enthalten, andererseits auch welche, die die Wörter allgemein durch Artikel oder News beschreiben.

Thesen Während meiner Arbeit verfolge ich folgende Forschungsansätze:

(1) Durch Worteinbettungen und Kosinusähnlichkeit kann man erkennen, ob benachbarte Textsegmente zusammen eine argumentative Aussage bilden.

(2) Durch Worteinbettungen und Betrachtung der Wortvektoren innerhalb eines Segments kann man beurteilen, ob ein Segment argumentativ ist oder nicht.

(3) Durch Worteinbettungen und Kosinusähnlichkeit als kontextuelle Features kann man Relationen zwischen 'Claim' und 'Premise' in argumentativen Texten erkennen.

(4) Durch einen größeren Trainingskorpus und damit einhergehenden größeren Wortschatz lassen sich auch bessere Ergebnisse mit Word2Vec erzielen.

(5) Durch das Extrahieren bestimmter Bestandteile aus Sätzen, wie Hauptwörter oder Nominalphrasen, lassen sich aussagekräftigere Kosinusähnlichkeitswerte berechnen.

(6) Ab einem bestimmten Kosinusähnlichkeitswert kann man entscheiden, ob eine Relation zwischen Segmenten bzw. Sätzen vorliegt oder nicht.

Zuerst soll der allgemeine Ablauf der automatischen argumentativen Analyse erläutert werden, welche Probleme auftreten und wie diese bisher gelöst wurden (Kapitel 2). Danach wird kurz auf die verwendete Software (Kapitel 3) und Datensätze (Kapitel 4) eingegangen, mit besonderen Augenmerk auf das Framework Word2Vec und die Training- und Klassifikationsdaten. Im Folgenden werden die Experimente beschrieben und die Resultate der Experimente aufgezeigt (Kapitel 5). Die Thesen (1) bis (3) sollen dabei anhand der Ergebnisse der Experimente 1 bis 3 nachgewiesen werden. Die Thesen (4) bis (6) hingegen sollen beim Vergleich der Word2Vec-Trainingsdaten, den Filterungsarten der Klassifikationsdaten und der besten Ergebnissen der Klassifikationsverfahren beurteilt werden. Zum Schluss werden die Ergebnisse diskutiert und auf weiterführende Verbesserungen eingegangen (Kapitel 6).

Kapitel 2

Die Argumentationsanalyse

In diesem Kapitel soll auf den klassischen Ablauf der Argumentationsanalyse eingegangen werden und welche Probleme in den einzelnen Schritten entstehen, die ich versuche in meiner Arbeit zu lösen. In Abschnitt 2.2 werden verwandte Arbeiten besprochen, die ebenfalls versucht haben durch Worteinbettung erste Ergebnisse in der automatischen Argumentationsanalyse zu erzielen.

2.1 Ablauf und Problemstellungen

Angelehnt an der 'Manual Argument Analysis' (Lawrence et al. [2014]) und eigenen Arbeitsverfahren, kann man die Argumentationsanalyse in vier Schritte einteilen.

1. Textsegmentierung
2. Argumentativ/nicht argumentativ
3. Einfache Strukturierung
4. Spezifischere Strukturierung

In den folgenden Abschnitten soll auf die einzelnen Schritte sowie ihre Problemstellungen eingegangen werden. Vorher sei zu erwähnen, dass die Schritte (2) bis (4) nicht völlig voneinander getrennt betrachtet werden sollten, sondern diese für optimale Ergebnisse eigentlich iterativ mehrmals durchgeführt werden müssten. Das liegt daran, da Erkenntnisse aus den Folgeschritten sich rückwirkend auf die Klassifizierung der vorherigen Schritte auswirken können. Dazu mehr in den jeweiligen Abschnitten.

2.1.1 Textsegmentierung

Bei dem ersten Schritt wird zunächst der argumentative Text in Segmente eingeteilt, die in den folgenden Schritten klassifiziert werden. In der Arbeit 'Approaches to Automatic Argumentative Unit Segmentation' (Kiesel [2016]) wurden bereits über 95% der Segmentgrenzen richtig erkannt, allerdings wurden auch neue Segmentgrenzen eingefügt, was zu Problemen in den folgenden Schritten führen kann. Die wirkliche Übereinstimmung beträgt deshalb nur ungefähr 75%. In vielen Verfahren betrachtet man die Segmente meist einzeln und will diese in argumentativ oder nicht argumentativ klassifizieren. Besteht nun aber dieses Segment unnötigerweise aus vielen Teilsegmenten, so lassen sich die Informationen schwieriger zusammentragen bzw. verbinden. Dieses Problem kann man auch als 'Übersegmentierung' bezeichnen.
Ebenfalls wurde in 'Mining Arguments From 19th Century Philosophical Texts Using Topic Based Modelling' (Lawrence et al. [2014]) gezeigt, dass man mit manueller Segmentierung per Hand nur eine Übereinstimmung von 88.5% erreicht hat, welches deshalb als 'Gold Standard' gilt. Man geht davon aus, dass durch maschinelles Lernen kein besserer Wert zu erwarten ist. An folgender Abbildung 2.1 soll das Problem der Segmentierung noch einmal verdeutlicht werden.

1 **It is always said**
2 **that**
3 **competition can effectively promote the development of economy.**

Abbildung 2.1: Beispiel für die Segmentierung eines Satzes.

Hier kann man erkennen, dass der Satz 'It is always said that competition can effectively promote the development of economy.' in drei Segmente aufgespalten wurde, die jeweils eine bestimmte Bedeutung haben. Während das dritte Segment die argumentative Aussage beinhaltet, enthält Segment 1 nur eine Einleitungsphrase und Segment 2 ein Indikatorwort. Wichtig sollte hierbei sein, das dritte Segment nicht weiter aufzuspalten, da es sonst im späteren Verlauf der Analyse schwierig wird dieses noch richtig zu klassifizieren.
Das erste Experiment in dieser Arbeit soll sich mit dem erwähnten Problem der Übersegmentierung beschäftigen. Die meisten Segmentgrenzen werden zwar mit Hilfe von Kommas und Satzgliedern erkannt, aber es werden auch oft falsche Grenzen eingefügt. Diese falschen Grenzen sollen durch Worteinbettung identifiziert werden, um die Segmentierung zu verbessern.

2.1.2 Argumentativ/nicht argumentativ

Nachdem der argumentative Text segmentiert wurde, geht es nun darum die Segmente in argumentativ oder nicht argumentativ zu klassifizieren. Dabei kann ein ganzer Satz als argumentativ gelten oder nur ein Segment von diesem. Ein Segment gilt dann als argumentativ, wenn der Leser ihm eine wichtige Information in Bezug auf die Argumentation entnehmen kann, ohne dazu ein weiteres Segment lesen zu müssen. Anhand von bestimmten Indikatoren, die meist vor dem entsprechendem Segment stehen, lassen sich viele Segmente bereits als Träger von Aussagen identifizieren. Ob diese aber auch argumentativ in Bezug auf das Thema des Gesamttextes sind, zeigt sich meist erst in den weiteren Schritten der argumentativen Analyse.
In dieser Arbeit wird als zweites Experiment mit Worteinbettung versucht ein Segment als argumentativ oder nicht argumentativ zu klassifizieren. Hierbei wird die bereits vorhandene Segmentierung aus dem ersten Schritt der argumentativen Analyse verwendet. Als weitere Features, um die Ergebnisse zu vergleichen und zu verbessern, werden ebenfalls die Positionsmerkmale der Segmente innerhalb des Textes analysiert.

2.1.3 Einfache Strukturierung

Hat man alle argumentativen Segmente im zweiten Schritt der argumentativen Analyse herausgefiltert, geht es nun darum die Beziehungen zwischen ihnen zu erkennen. Geht man nach der Rhetorical Structure Theory (Mann and Thompson [1988]) und 'Argumentative Text as Rhetorical Structure: An Application of Rhetorical Structure Theory' (Azar [1999]), so kann man die gefilterten Segmente in 'Nucleus' oder 'Satellite' klassifizieren. Nucleus ist dabei die Hauptaussage, also die Meinung des Textverfassers, und 'Satellite' der Beitrag zu dieser Aussage, um diese zu unterstützen. In jüngsten Forschungsarbeiten bezüglich argumentativen Texten wie in 'Annotating Argument Components and Relations in Persuasive Essays' Stab and Gurevych [2014] werden aber lieber die Begriffe 'Claim' und 'Premise' verwendet, welche eine spezifischere Namenskonvention in Hinblick auf Argumentationen sind und nicht allgemein für alle Texte. In dieser Arbeit werden ebenfalls die letzten zwei genannten Begriffe verwendet. Folgendes Beispiel in Abbildung 2.2 soll die Beziehungen zwischen Claim und Premise aufzeigen.

Zu sehen ist, dass die erste Aussage als Claim, also Kernaussage klassifiziert wurde, während die anderen beiden Aussagen Premise sind, die die erste Aussage unterstützen. Hieraus kann man schon erkennen, dass meistens eine 1:n oder zumindest 1:1 Beziehung zwischen Claim und Premise vorliegt, damit eine Kernaussage von ein oder mehreren Beispielen untermauert wird.

A1(Claim)	**we should attach more importance to cooperation**
A2(Premise)	**without the cooperation, there would be no victory of competition**
A3(Premise)	**a more cooperative attitudes towards life is more profitable in one's success**

Abbildung 2.2: Beispiele für Aussagen, die als 'Claim' und 'Premise' klassifiziert wurden.

Im dritten Experiment meiner Arbeit versuche ich mit Worteinbettung und Kosinusähnlichkeit die Relationen zwischen argumentativen Einheiten zu erkennen, um diese als Claim und Premise klassifizieren zu können.

2.1.4 Spezifischere Strukturierung

Bei dem letzten Schritt der argumentativen Analyse geht es um die spezifischere Klassifizierung der im letzten Schritt erwähnten Premise. Diese können entweder als 'support' oder 'attack' klassifiziert werden, je nachdem ob das Beispiel die Kernaussage unterstützt oder angreift. Dies lässt sich teilweise schon an Diskurs-Indikatoren vor oder innerhalb einer Premise erkennen, durch unterstützende Wörter wie 'therefore' oder 'because' bzw. konfliktaufrufende Wörter wie 'however' oder 'but' (Webber et al. [2012]).
In dieser Arbeit wird dieser Schritt nicht in Betracht gezogen. Nicht, weil es nicht möglich wäre auch hier mit Worteinbettung Ergebnisse zu erzielen, sondern weil in den Klassifikationsdaten die Anzahl der als 'attack' klassifizierten Premise zu klein ist, um sinnvolles maschinelles Lernen anwenden zu können.

2.2 Verwandte Arbeiten

In diesem Abschnitt wird auf Arbeiten eingegangen, die sich ebenfalls mit der argumentativen Analyse und Worteinbettungen beschäftigt haben. Dazu lässt sich aber vorweg sagen, dass sich bisher nur wenige Forschungen mit beiden Themen gleichzeitig auseinander gesetzt haben.

Um Relationen in argumentativen Texten besser erkennen und klassifizieren zu können, ist es ein guter Ansatz, zuerst ein Topic-Model aufzustellen, um verschiedene Arten von Relationen in Themen zu unterscheiden. Diesen Ansatz verfolgen die Arbeiten 'Extracting Argument and Domain Words for Identifying Argument Components in Texts' (Nguyen and Litman [2015]) und 'From Argumentation Mining to Stance Classification' (Sobhani et al. [2015])

und wird im Kapitel 6.2.3 zum Thema Word2Vec in Bezug auf 'Doc2Vec' wieder aufgegriffen. In der ersten Arbeit bezüglich Domain Words werden für jedes 'Genre' andere Indikatoren gelernt, um bessere Ergebnisse in der Klassifikation von argumentativen Einheiten zu erzielen und ebenfalls Relationen besser zu erkennen. Man konnte feststellen, dass Indikatoren für argumentative Einheiten je nach Domain (also Thema des argumentativen Textes) anders ausfallen können, und deshalb auch getrennt voneinander gelernt werden sollten. Die zweite Arbeit bezüglich Stance Classification hingegen, verwendet auch Domain Words für die jeweiligen Texte (so kann zum Beispiel das Thema eines argumentativen Textes sein 'Financial benefit of the study') und schaut dann, wie stark die Meinung zu diesem Thema in dem Text vertreten ist, indem sie mit 'Strongly For', 'For', 'Against' und 'Strongly Against' für die jeweiligen argumentativen Segmente annotiert wird. Dadurch lässt sich ebenfalls ein Modell erstellen, mit dem man Vorhersagen zu neuen Argumenten machen kann. Die Genauigkeit dabei argumentative Segmente zu klassifizieren, lag hier bei 79%, wobei Premise mit 84% deutlich besser erkannt wurden als zum Beispiel Major Claim und Claim mit rund 52%.

Eine Arbeit, die sich tatsächlich damit beschäftigt hat mit Worteinbettung durch Word2Vec Ergebnisse in der argumentativen Analyse zu erzielen, ist 'Argument Extraction from News' (Sardianos et al. [2015]). Im Gegensatz zu meiner Arbeit, fanden allerdings alle Experimente nur mit griechischen Trainings- und Klassifikationsdatensätzen statt. Dennoch konnte ein Resultat von dieser Arbeit auf meine übertragen werden. So hat sich herausgestellt, dass Modelle, die aus Trainingsdaten von News-Seiten oder Blogs stammen, deutlich bessere Wortrelationen vorher sagen können, als von Kommentaren aus Facebook oder Twitter. Dies liegt daran, weil die zuerst genannten Quellen bessere Strukturen aufweisen und mehr auf Rechtschreibung achten, als die zuletzt genannten, die meist nur kurze rechtschreibfehlerbehaftete Einzeiler sind. Da es schwierig ist aus solchen Quellen sinnvolle semantische Relationen zu lernen, habe ich für meine Arbeit auch nur Texte aus wohlformulierten Quellen verwendet (siehe Kapitel 4 'Datensätze'). Der griechische annotierte Korpus für die Klassifikationsdaten besteht aus 4524 Segmenten, wovon 1191 als argumentativ (also 'Claim' oder 'Premise') annotiert sind. Ziel bei den Experimenten hier war es nicht, die Relationen zwischen den argumentativen Einheiten zu erkennen, sondern nur die Segmente selber als argumentativ bzw. nicht argumentativ zu klassifizieren (entspricht Experiment 2 meiner Arbeit). Obwohl hier sehr nüchterne Ergebnisse entstanden sind, konnte bezüglich der Kontextfenstergröße eine weitere Erkenntnis erzielt werden. Die Fenstergröße gibt an, wie groß der Bereich sein soll, indem Relationen zwischen Wörtern betrachtet werden. Diese hat man für die Experimente auf 0, 2 oder 5 festgelegt, wobei mit

größeren Fenster, sich auch die Ergebnisse verbessert haben. Die Parameterwahl von Word2Vec für meine Experimente diskutiere ich in Kapitel 3.2.1 'Die Parameter von Word2Vec'.

Kapitel 3

Verwendete Software

In diesem Kapitel sollen die verschiedenen Programme erklärt werden, mit denen die Daten verarbeitet, die Experimente durchgeführt und die Ergebnisse evaluiert wurden. Um die Daten vorzubereiten und bestimmte Bestandteile zu extrahieren, wurden zuerst die 'Stanford CoreNLP Tools' verwendet (Kapitel 3.1). Die Worteinbettung zur Durchführung der Experimente wurde mit 'Word2Vec' vorgenommen (Kapitel 3.2). Funktionsweise und Parameter dieses Frameworks werden dabei in weiteren Unterkapiteln genauer erläutert. Zum Schluss wird die Machine Learning Software 'Weka' vorgestellt (Kapitel 3.3), mit der die entstanden Resultate aus den Experimenten durch Klassifizierungsverfahren evaluiert wurden.

3.1 Datenextrahierung mit den Stanford CoreNLP Tools

Bei diesem Framework von der Stanford Natural Language Processing Group[1] handelt es sich um eine Kollektion von Werkzeugen für Java, mit denen man Sätze auf ihre Bestandteile hin analysieren kann. Zum einen wurden die Werkzeuge benutzt, um aus den Klassifikationsdaten, die entweder aus längeren Segmenten oder ganzen Sätzen bestehen, Nomen oder Nominalphrasen zu extrahieren. Siehe dazu das Beispiel in Abbildung 3.1.

Die Nomen sind dabei Subjekte und Objekte eines Satzes, während Nominalphrasen dies ebenfalls sind, aber gegebenenfalls mit dazugehörigen Artikeln und Adjektiven, welche das Nomen beschreiben. Im gezeigten Beispiel sind 'Baum' und 'Blitz' beides Nomen, während als Nominalphrase für Baum noch die Wörter 'der' und 'alte' dazu kommen. Durch dieses Verfahren soll der These 5 nachgegangen werden, ob man durch das Extrahieren bestimmter

[1] `http://nlp.stanford.edu/software/corenlp.shtml`

1 **Satz:** Der alte Baum wurde vom Blitz getroffen.
2 **Nomen:** Baum, Blitz
3 **Nominalphrasen:** der alte Baum, Blitz

Abbildung 3.1: Beispiel für Extrahierung von Nomen und Nominalphrasen aus einem Satz.

Bestandteile aus den Klassifikationsdaten, bessere Werte für die Experimente berechnen kann.

Zur Erweiterung der Experimente, wurden die Tools auch benutzt, um Koreferenzen aufzulösen und Lemmatisierung durchzuführen. Bei der Koreferenzauflösung wird geschaut, ob sich eine Referenz in einem Satz befindet, wie zum Beispiel 'they', 'them' oder 'she', die sich auf eine vorher erwähnte Person oder Sache bezieht. Ist dies der Fall, so wird die Referenz durch den passenden Bezeichner für die Person oder Sache ersetzt. Dadurch hat man den Vorteil, dass auch in den Prämissen, in denen der Referent nicht wiederholt wird, dieser trotzdem in der Aussage steht. Gegebenenfalls können dadurch auch bessere Ergebnisse erzielt werden, weil die Relationen besser erkannt werden.
Die Lemmatisierung hingegen sollte beim Lernen der Wortvektoren aus den Trainingsdaten helfen, um bessere Modelle aufzustellen und ebenfalls zur besseren Erkennung der Wörter in den Klassifizierungsdaten beitragen, wenn zumindest dieses Verfahren bei beiden Korpora gleichermaßen durchgeführt wurde. Bei der Lemmatisierung werden nach Fällen angepasste Wörter wieder auf ihre Ursprungsform zurückverwandelt. Beispiele hierfür wären das Verb 'plays', welches in seinen Infinitiv 'play' umgeändert wird, oder das Substantiv 'moralities' in seiner Pluralform, welches in seine Singularform 'morality' abgeändert wird. Wörter werden damit für die Modelle nur einmal gelernt und nicht in verschiedenen Formen, je nach Fallanpassung. Im Gegensatz zur Lemmatisierung, gibt es auch noch das etwas bekanntere aber auch aggressivere 'Stemming' (auf Deutsch: zum Stamm reduzieren). Dieses Verfahren wird in meiner Arbeit nicht verwendet, da Wörter, egal ob Verb, Substantiv oder Adjektiv, gleichermaßen zu einem Wortstamm umgewandelt werden. Wörter wie 'moralize' und 'moralities', die Verb und Substantiv sind, würden dann beide in 'moral' umgewandelt werden. Da es, wie oben erwähnt, ein Vorgehen meiner Arbeit ist auch Nomen und andere Satzbestandteile zu extrahieren, wurde dieses Verfahren nicht durchgeführt, da es Wortformen zerstört und eventuell auch relationale Eigenschaften zwischen Wörtern aufhebt.

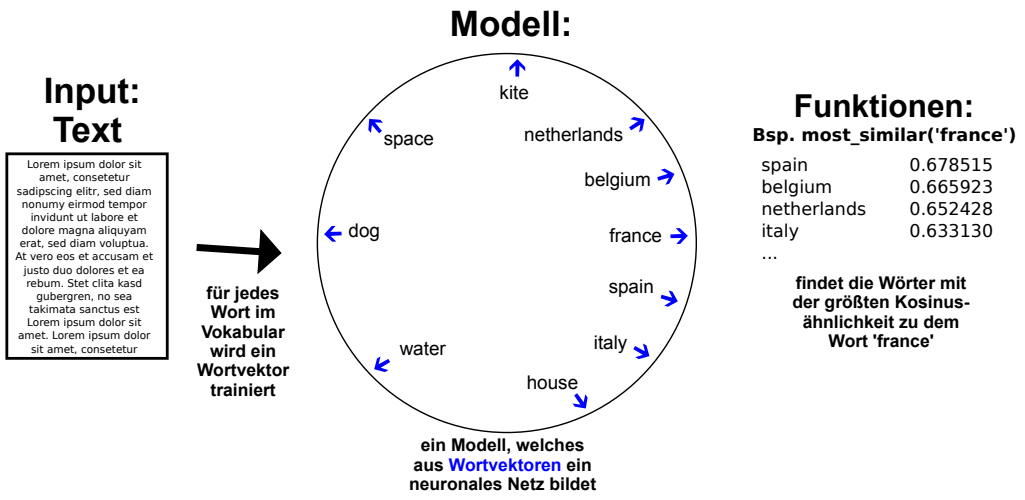

Abbildung 3.2: Ablauf von Word2Vec

3.2 Worteinbettung mit Word2Vec

Word2Vec (Mikolov et al. [2013]) ist ein Framework, welches aus einen Textkorpus ein neuronales Netz aus Wortvektoren erstellt, welches für verschiedene Aufgaben in der maschinellen Sprachverarbeitung verwendet werden kann. Das Ziel dabei ist es zu erkennen, dass Wörter, die in einem gleichen Kontext verwendet werden, aus ähnlichen Vektoren bestehen und damit Relationen aufweisen. Aus den Wortvektoren kann ein Modell erstellt werden, welches mit dem Framework verschiedene Funktionen mit sich bringt. In der Abbildung 3.2 wird der allgemeine Ablauf von Word2Vec bildlich dargestellt und eine Funktion gezeigt.

Eine wichtige Funktion in Word2Vec, die ich in meiner Arbeit verwende, ist die Berechnung der Kosinusähnlichkeit zweier Wörter oder Wortgruppen, mit der sich ermitteln lässt, wie nahe sich diese Wörter in einer Relation stehen. Dieser Wert kann zwischen -1 und 1 liegen. Je größer der Wert ist, desto größer ist die Beziehung zwischen den zwei Wörtern, wobei 1 nur dann erreicht werden kann, wenn man die Kosinusähnlichkeit mit einem Wort zu sich selber berechnet.
Im folgenden werden die Parameter von Word2Vec zum Erstellen der Wortvektoren genauer erklärt. In einem weiteren Abschnitt wird ein Augenmerk auf gewisse Charakteristika der Trainingsdaten gelegt und warum die Gensim Implementierung von Word2Vec verwendet wurde.

3.2.1 Die Parameter von Word2Vec

Um mit Word2Vec Wortvektoren zu erstellen und damit ein Modell zu trainieren, muss man vorher bestimmte Parameter festlegen. Diese sollen nun kurz erläutert werden.

sg SG gibt an, ob man das Modell im Skip-Gram (1, also true) oder alternativ im Bag-of-Words Verfahren (0, also false) erstellen will. Der Hauptunterschied zwischen beiden Verfahren liegt darin, dass im Skip-Gram Modell der gesuchte Wortvektor der Input Layer ist, während die Kontextworte im Output Layer sind und über einen Hidden Layer und zwei Matrizen berechnet werden. Im Bag-of-Words Modell sind diese In- und Output Layer vertauscht. Nach Mikolov et al. erzeugt das erste Modell durchgängig bessere Resultate, weshalb ich dieses auch in meiner Arbeit verwende.

size Die Size gibt an, in wie vielen Vektordimensionen ein Wort abgebildet werden soll. Diesen Wert sollte man davon abhängig machen, wie groß der Trainingskorpus ist, damit möglichst alle Relationen von einem Wort zu jedem anderen Wort ausgedrückt werden können, ohne Informationen einbüßen zu müssen. Für kleine Trainingsets wird deshalb eine Vektordimension von ca. 100 empfohlen, während größere Korpora öfters mit einer Size von 300 bis 1000 erstellt werden.

window Window ist die maximale Fenstergröße. Das Fenster wandert dabei über den gesamten Textkorpus um die Vektoren für die Kontextworte zu einem bestimmten Wortvektor zu bilden. Das Fenster wird kleiner, wenn es sich einem Satzendpunkt oder Umbruch nähert. Bei bereits zusammengefassten Stichpunkten, die Informationen kompakter enthalten, wird eine Fenstergröße von 3∼5 empfohlen, während bei normalen Texten eine Fenstergröße von 5∼8 verwendet wird, um Relationen in größeren Grenzen auch um Stoppwörtern zu erkennen.

min_count Normalerweise wird zu jedem Wort in einem Korpus ein Wortvektor erstellt. Da man aber nicht unbedingt jedes Wort in das endgültige Vokabular mit aufnehmen will, kann man die Häufigkeit jedes Wortes zuerst zählen lassen und nur diese in die Berechnung mitaufnehmen, die eine Mindesthäufigkeit erfüllen. Damit wird versichert, dass nur Vektoren zu Wörtern aufgestellt werden, die auch über genügend Präsenz im Korpus verfügen und damit Kontextinformationen enthalten. Je nach Größe des Trainingsdatensets wurde dieser Wert gegebenenfalls erhöht, um nur häufig präsente Wörter in das Modell mit aufzunehmen, aber auch um eventuelle Fehlformatierungen

(die z.B. in Wikipedia durchaus vorkommen können) aus dem Vokabular zu entfernen.

sample Die Samplerate gibt den Threshold an, ab wann Wörter mit zu hoher Frequenz wieder gedownsampled werden sollen. Als Standardwert wird hier e^{-5} verwendet und auch in meinen Experimenten nicht verändert.

hs Hierarchical soft-max (Morin and Bengio, 2005; Mnih and Hinton, 2009) kann für das Training der Wörter aktiviert werden. Dabei wird ein baumartiger Durchlauf auf den Output Layern durchgeführt, der die Effizienz des Trainings von $\mathcal{O}(N)$ auf $\mathcal{O}(\log N)$ verbessert. Da dies zu keinen weiteren Nachteilen führt, wurde diese Option in meinen Experimenten standardmäßig aktiviert.

negative Durch diesen Wert gibt man an, wie viele Negative Sampling Layer man für das Training erstellen will. Das Negative Sampling ist dabei an das numerische Gradientenverfahren angelehnt und soll ebenfalls das Training effizienter machen. Das Grundprinzip liegt darin, nicht den Ort zu erraten, wo ein Wort sein könnte, sondern wo es nicht sein könnte. Ist der Hierarchical soft-max aktiviert, findet kein Negative Sampling statt. Nach Mikolov et al. eignet sich Hierarchical soft-max mehr für vereinzelte Wörter, während Negative Sampling eher bei häufigen Wörtern bessere Vektoren aufstellt. Da die Schlagwörter in meinen Trainingsdatensets immer nur in einem Artikel oder Essay zu finden sind und gelernt werden sollen und nicht im gesamten Set, habe ich hier den Hierarchical soft-max bevorzugt.

iter Mit der Anzahl an Iterationen wird angegeben, wie oft der Trainingsprozess über den gesamten Korpus wandern soll. Hierbei wird die Reihenfolge der Sätze bzw. Paragraphen mit jeder Iteration variiert. Dies hat zur Folge, dass die Wortvektoren besser gelernt werden, natürlich auf Kosten der Zeit. Um möglichst gute Modelle aufzustellen, habe ich hier mit 20 eine relativ hohe Anzahl an Iterationen für jedes Trainingsdatenset festgelegt.

3.2.2 Ähnlichkeitsverteilung und Aufbau des Trainingskorpus

Neben den genannten Parametern hat Word2Vec noch wichtige Eigenschaften, welche in 'Measuring Word Significance using Distributed Representations of Words' (Schakel and Wilson [2015]) analysiert wurden: die Ähnlichkeitsverteilung und die Kontextanalyse.

Zum einen sollte nach der Definition von Word2Vec, die Ähnlichkeitsverteilung aller Wortpaare in einem Modell bei ungefähr 0 liegen, d.h. man sollte auf diesen Wert kommen, wenn man die Kosinusähnlichkeit aller möglichen Wortpaare eines Modells miteinander addiert. In der Praxis hat sich dies aber nicht immer als wahr herausgestellt. Oftmals liegt der zusammengerechnete Wert eher bei 0.2, was dann zu Stande kommt, wenn der Korpus, der für das Training verwendet wurde, zu klein ist. Eine solche Schwankung in der Ähnlichkeitsverteilung kann ebenfalls zu Ungenauigkeiten in der Berechnung der Kosinusähnlichkeit führen. Welche Ähnlichkeitsverteilung die Modelle in meinen Experimenten hatten, kann im Kapitel 6.2.1 zur Diskussion der Trainingsdatensets angeschaut werden, bei der ich noch einmal auf dieses Thema eingehe.

Weiterhin ist Word2Vec darauf ausgelegt, Relationen zu anderen Wörtern zu erkennen. Problematisch wird dies aber, wenn ein Wort in verschiedenen Kontexten verwendet wird. So kann zum Beispiel das Wort 'germany' in einem politischen, wirtschaftlichen oder geografischen Zusammenhang genannt werden, aber je nachdem verschiedene andere Wörter in Relation haben. Wenn ein Wort in zu vielen Kontexten genannt wird, die von den Wörtern im Modell vielleicht sogar bisher gegenteilig trainiert wurden (Vektor zeigt in die entgegengesetzte Richtung), führt dass dazu, dass dieser Wortvektor 'schwammig' wird und seine genaue einzigartige Bedeutung verliert. Um verschiedene Kontexte für ein Wort zu trainieren, wurde deshalb von Quoc V. Le und Tomas Mikolov 2014 das weiterführende Framework 'Paragraph2Vec' (Le and Mikolov [2014]) entwickelt. Dieses Framework erstellt neben den Wortvektoren noch Vektoren für die einzelnen Kontexte (wenn man diese vorher im Trainingsdatenset so organisiert hat). Auf 'Doc2Vec', die Gensim-Bezeichnung für 'Paragraph2Vec', wird noch einmal in Kapitel 6 zur Diskussion eingegangen.

3.2.3 Die Python Gensim Implementierung von Word2Vec

Neben der normalen C-Implementierung von Word2Vec gibt es noch weitere Implementierungen, die auch auf der offiziellen Webseite[2] von Word2Vec referenziert werden. Eins der meist verwendeten Frameworks ist hierbei die Gensim Implementierung[3] in Python. Diese bringt in Gegensatz zur Standardimplementierung und anderen Frameworks verschiedene Vorteile mit sich.

[2] `https://code.google.com/p/word2vec/`
[3] `https://radimrehurek.com/gensim/models/word2vec.html`

Zum einen lassen sich bestimmte Parameter während der Trainingsphase variieren. So kann man, falls dies gewollt ist, nach jeder Iteration zum Beispiel den Alpha-Wert anpassen, der für die Lernrate verantwortlich ist (wie schnell ein neuer Wert zu einen alten eingerechnet werden soll). Oder man kann die Fenstergröße verändern, wenn man nicht mehr Sätze als Training übergibt, sondern nur kurze Stichpunkte. Von der Änderung der Parameter während des Trainings, habe ich bei meinen Experimenten keinen Gebrauch gemacht.
Ein weiterer Vorteil ist das Vorhandensein vieler bereits eingebundener Funktionen, die es in der ursprünglichen C-Implementierung noch nicht gab und andere Implementierungen in Java oder sonstigen Plattformen auch fehlen. Hier gibt es zum Beispiel die Funktion 'n_similarity', welche längere Wortketten, also Segmente oder Sätze, entgegen nehmen kann, um zwischen ihnen die Kosinusähnlichkeit zu berechnen.
Zum Schluss sei noch die Integration und Kompatibilität zu Paragraph2Vec (in Gensim als 'Doc2Vec'[4] bezeichnet) genannt, bei der man die gleichen Funktionen anwenden kann wie in der Word2Vec Implementation. Die Trainingsprozedur ist dabei ebenfalls ähnlich gelöst und Parameter können variiert werden. Andere Implementationen haben dieses erweiterte Framework noch nicht integriert bzw. stabil zum Laufen gebracht.

3.3 Maschinelles Lernverfahren mit Weka

Um die mit Word2Vec entstandenen Wortvektoren oder berechneten Kosinusähnlichkeiten zu klassifizieren, wurde die Data Mining Software Weka 3[5] von der University of Waikato verwendet. Die entstandenen Daten aus den Experimenten wurden als .arff-Datei umgewandelt und Weka übergeben, um sie mit verschiedenen integrierten Klassifizierungsverfahren zu evaluieren.

Als Klassifizierungsverfahren in Weka wurden die Verfahren 'OneR' und 'J48' ausgewählt. OneR ist ein Verfahren, welches 1993 von R.C. Holte entwickelt wurde. Es nimmt sich das Feature mit der größten Aussagekraft und falls dieses ein Zahlenwert ist, versucht es diesen für die besten Ergebnisse zu diskretisieren. Anders formuliert bedeutet das, dass in einem bestimmten Zahlenraum Bereiche aufgestellt werden, die dann (falls es zum Beispiel nur zwei Klassen gibt) als true oder false klassifiziert werden. Da die Kosinusähnlichkeit in den Experimenten 1 und 3 das einzige Attribut ist, wird nur dieses klassifiziert und in Bereiche aufgeteilt, da es ein Zahlenwert ist.
J48 hingegen ist ein Klassifizierungsverfahren mit dem man einen Entschei-

[4] `https://radimrehurek.com/gensim/models/doc2vec.html`
[5] `http://www.cs.waikato.ac.nz/ml/weka/`

```
OneR Classifier model:
        cd <  -0.5      -> false
        cd <   0.0      -> true
        cd <   0.5      -> false
        cd >=  0.5      -> true

J48 Classifier model:
        cd <   0.2      -> false
        cd >=  0.2      -> true
```

Abbildung 3.3: Beispiele für Klassifizierungsmodelle erstellt mit OneR und J48. Während OneR mehrere Entscheidungsbereiche diskretisiert, versucht J48 mit einer Entscheidung den kleinsten Fehler zu erreichen.

dungsbaum aufstellt. Der Algorithmus für den C4.5 Entscheidungsbaum, der dabei erstellt wird, wurde von Ross Quinlan 1993 entworfen. Für jedes Attribut wird hier eine binäre Entscheidung getroffen. Für die Kosinusähnlichkeit wird eine Grenze festgelegt, bei welchen Wert etwas als 'false' klassifiziert wird und ab welchen Wert etwas als 'true' gilt. Der Unterschied zwischen beiden Algorithmen soll in 3.3 gezeigt werden.

Um neben den Kosinusähnlichkeitswerten, die aus Word2Vec entstanden sind, einen Vergleichswert zu haben, wurde ebenfalls ein 'Bag-of-Words' Verfahren in Weka angewandt. Dazu wurden mit dem Filter 'StringToWordVector' alle vorkommenden Wörter als Features umgewandelt und geprüft, ob diese in den Segmenten bzw. Sätzen vorkommen. Ist dies der Fall, ist der Wert true oder false für das jeweilige Wort. Dadurch wird nur das Vorkommen von Wörtern betrachtet, aber nicht die kontextuelle Eigenschaft wie bei Word2Vec. So wird mit 'J48' klassifiziert, dass wenn zum Beispiel das Wort 'because' in einem Segment oder Satz steht, dieses Segment bzw. dieser Satz zu einem bestimmten Prozentsatz argumentativ ist oder nicht. Ziel ist es, die Ergebnisse des Bag-of-Words Verfahren mit den Ergebnissen der Worteinbettung zu schlagen.

Kapitel 4

Datensätze

In diesem Kapitel wird auf die verwendeten Daten für die Experimente eingegangen, die sich mit der Worteinbettung und dem Schließen auf kontextuelle Eigenschaften befassen. Die Daten werden dabei einmal in Trainingsdaten und Klassifikationsdaten unterschieden. Die Trainingsdaten sind Rohtexte von verschiedenen Quellen, aus denen mit Word2Vec die Modelle erstellt wurden. Die Klassifikationsdaten hingegen sind annotierte Korpora, an denen getestet werden soll, ob die Modelle gute Ergebnisse bezüglich kontextueller Eigenschaften liefern. Damit alle Datensätze kompatibel und vergleichbar miteinander sind, wurden bestimmte Umformatierungen vorgenommen, wie das Ändern auf Kleinschreibung oder das Entfernen von Sonderzeichen. Für die Daten, die in den späteren Kapiteln immer wieder Erwähnung finden, wurden die neuen Abkürzungen 'uppsala', 'text8', 'wiki14' und 'news' für die Trainingsdaten und daraus entstandenen Modelle eingeführt, und 'AAEC' und 'AMC' für die Klassifikationsdatensätze. In den folgenden Unterkapiteln werden diese Abkürzungen erläutert.

4.1 Trainingsdaten für Word2Vec

Für die Experimente mit Word2Vec wurden verschiedene Trainingsdatensets rausgesucht, die in diesen Abschnitt erläutert werden. Die entstandenen Modelle haben verschiedene Wortvektoren, da die kontextuellen Eigenschaften jedes Wortes je nach Trainingsdaten anders gelernt wurden. Damit soll These 3 nachgegangen werden, dass Modelle, die durch ein größeres Trainingsdatenset erstellt wurden, in Bezug auf Erkennung von Kontexten bessere Ergebnisse erzielen. Alle Modelle, bis auf 'news', wurden mit der Python Gensim Implementation von Word2Vec erstellt. Die Parameter sg=1, hs=1, window=8, sample=1e-5 und iter=20 sind bei jedem Training der Modelle gleich geblieben. Nur die Parameter size und min_count wurden variiert, da diese vom

Wortschatz und der Größe des Trainingskorpus abhängig sind. Die Abkürzungen sg und hs geben an, ob das Modell per Skip-Gram erstellt wurde und dabei einen Hierarchical soft-max benutzt - zwei standardmäßig aktivierte Verfahren. Weiterhin wurde der Standardwert von sample beibehalten, der bewirkt, dass hochfrequente Wörter im Modell weniger dominieren. Da es sich bei jeden Trainingskorpus um Texte mit ganzen Sätzen handelt, wurde ein Mittelwert zwischen den empfohlenen 5 bis 10 Wörtern als window festgelegt. Dieser Wert müsste nur dann angepasst werden, wenn es sich bei einem Korpus nur um kurze Stichpunkte handelt oder andere vorstrukturierte Formen. Zuletzt wurde der Wert iteration auf 20 festgelegt, damit Word2Vec immer 20 mal den Korpus durchwandert, um die Wortvektoren möglichst genau zu trainieren. Im Normalfall werden standardmäßig nur 5 oder 10 Iterationen durchgeführt, da sich eine höhere Anzahl an Iteration nur negativ auf die Bearbeitungszeit auswirkt, aber positiv auf die Vektoren, wurde dieser Wert höher angesetzt. Für eine genauere Erklärung der einzelnen Parameter und deren Einflüsse, wird im Kapitel 3.2.1 'Die Parameter von Word2Vec' noch einmal genauer eingegangen. Im Folgenden werden die Trainingsdaten für die Modelle der Größe nach sortiert vorgestellt.

uppsala 'uppsala' ist ein Korpus mit englischen Essays von der schwedischen Uppsala University[1]. Er hat eine Größe von 7MB und umfasst 1.489 Essays mit 1.221.265 Wörtern, geschrieben von 440 unterschiedlichen Studenten. Dieses Trainingsdatenset wurde trotz seiner kleinen Größe, in Vergleich zu anderen Korpora, ausgesucht, weil es ausschließlich argumentative Texte enthält, ähnlich wie die Klassifikationsdaten. Für das Modell sollten also die passenden kontextuellen Eigenschaften der Wörter gelernt werden, die für die Vorhersage einer Argumentation wichtig sind. Wortschatz und Häufigkeit der Wörter fallen hier allerdings sehr gering aus. Das Modell wurde deshalb mit einer size von 100 erstellt, da mit geringerer Korpusgröße, auch weniger Vektoren von nöten sind, um alle Relationen zu lernen. Da viele Wörter nur wenige male pro Essay-Thema auftauchen, wurde auch nur ein min-count von 2 festgelegt, damit trotz der kleinen Größe des Korpus, fast jedes Wort darin als Wortschatz aufgenommen und trainiert werden kann. Im Folgenden werden die Parameter zum trainierten Modell noch einmal zusammengefasst:

```
sg=1, size=100, window=8, min_count=2,
sample=1e-5, hs=1, iter=20
```

[1] `http://www.engelska.uu.se/Forskning/engelsk_sprakvetenskap/Forskningsomraden/Electronic_Resource_Projects/USE-Corpus/`

text8 Bei dem 'text8'-Korpus handelt es sich um einen kleinen Textkorpus der standardmäßig oft im Zusammenhang mit Word2Vec verwendet wird, um möglichst schnelle und gute Ergebnisse zu erzielen. Er wird von Matt Mahoney[2] angeboten und beinhaltet die ersten Milliarden Zeichen als reine Artikeltexte von der englischen Wikipedia von 2006. Der Rohtext hat hierbei eine Größe von 100 MB. Das hier entstandene Modell, welches für verschiedene Word2Vec Experimente in anderen Forschungen verwendet wird, soll auch hier zeigen, ob es gute Ergebnisse erzielen kann. Gegebenenfalls soll es einen Vergleich bieten zu uppsala oder text8, ob der argumentative Inhalt mehr eine Rolle beim Training spielt oder die Größe des Wortschatzes und in welchen Umfang die Wortvektoren gelernt wurden. Aufgrund der Größe dieses Korpus, wurde hier eine size von 300 gewählt, um den größeren Wortschatz und Relationen zu lernen, als auch ein min_count von 25, um eventuelle sehr selten genannte Wörter in der Wikipedia von 2006 nicht in das Modell mit aufzunehmen. Im Folgenden werden die Parameter zum trainierten Modell noch einmal zusammengefasst:

```
sg=1, size=300, window=8, min_count=25,
sample=1e-5, hs=1, iter=20
```

wiki14 Bei wiki14 handelt es sich im Gegensatz zu text8 um einen aktuelleren Wikipedia Textkorpus. Die rohen Textdaten wurden hierbei von MTA SZTAKI Department of Distributed Systems[3] bereitgestellt und beinhalten alle reinen Artikeltexte von der englischen Wikipedia von 2014. Der Rohtext mit entfernten Sonderzeichen und spezieller Wikipedia-Formatierungen (wie bei Überschriften, Listen oder URLs) hat eine Größe von 5,1 GB. Mit diesen Korpus soll speziell These 4 nachgegangen werden, ob im Vergleich zu text8 bessere Ergebnisse erzielt werden konnten, weil das Trainingsdatenset viel größer ausfällt von 100MB auf 5,1GB. Dadurch wurde auch der min_count Parameter erhöht, da eventuelle selten auftauchende Wörter oder gar Fehlformatierungen so aus dem Wortschatz rausgefiltert werden. Der size-Wert von 300 wurde beibehalten, da sich der Wortschatz selber nur unwesentlich vergrößert hat und mit 300 schon einen recht hohen Wert besitzt für große Korpora. Im Folgenden werden die Parameter zum trainierten Modell noch einmal zusammengefasst:

```
sg=1, size=300, window=8, min_count=100,
sample=1e-5, hs=1, iter=20
```

news Bei 'news' handelt es sich um ein bereits fertig trainiertes Modell im BIN-Format. Es wurde mit der normalen C-Implementierung von Word2Vec

[2] `http://mattmahoney.net/dc/textdata`

[3] `http://dsd.sztaki.hu/`

erstellt und hatte als Rohdaten über 100 Milliarden Wörter von verschiedenen News-Artikeln, die von freebase[4] zur Verfügung gestellt und von den Word2Vec-Entwicklern zu Wortvektoren umgewandelt wurden[5]. Anzumerken sei hier, dass Stoppwörter aus den Trainingsdaten entfernt wurden und deshalb im Wortschatz des fertig trainierten Modells nicht auftauchen. Dies sollte sich in der Theorie positiv auf die Ergebnisse auswirken, da Stoppwörter in der Regel durch ihr ständiges auftauchen, keine sinnvollen Wortvektoren mit relationaler Aussagekraft hervorbringen. Weiterhin ist von der anbietenden Webseite zu entnehmen, dass ebenfalls zur Erstellung des Modells das Skip-Gram Verfahren verwendet wurde und man eine size von 1000 Vektoren pro Wort verwendet hat. Über die anderen Parameter und der ursprünglichen Größe des Trainingskorpus (vor und nach Entfernung der Stoppwörter) erhält man keine Informationen.

4.2 Die Daten für die Klassifikation

Um die späteren Modelle validieren zu können, wurden zwei Korpora bezüglich Argumentation ausgesucht: AAEC (Stab and Gurevych [2014]) und AMC (Carstens and Toni [2015]). Diese wurden auf ihre argumentativen Bestandteile annotiert und eignen sich deshalb gut für die Experimente in dieser Arbeit. Während bei AAEC genaue Segmente innerhalb eines Textes auf Argumentation annotiert wurden, wurden bei AMC vollständige Sätze als argumentativ eingestuft. Weitere Informationen zu den Namen, der Herkunft der Daten sowie der genauen Annotation folgen in den nächsten Abschnitten.

AAEC Bei dem ersten Korpus wurden die Rohdaten von 90 argumentativen Aufsätzen von der Seite `www.essayforum.com` verwendet. Jeder der 90 Aufsätze behandelt dabei ein anderes Thema, wie zum Beispiel ob Zeitungen am Aussterben sind oder ob reiche Länder armen Ländern helfen sollten. Diese wurden 2014 von Stab und Gurevych (Stab and Gurevych [2014]) annotiert, wobei die argumentativen Segmente die Annotierung 'Major Claim', 'Claim' oder 'Premise' bekamen (ab hier als 'Ground Truth' bezeichnet). Die Kurzbezeichnung 'AAEC' steht für Argument Annotated Essays Corpus und der Korpus wurde mit dem Brat Annotation Tool annotiert. Bei meinen zweiten Experiment bezüglich der Klassifizierung in argumentativ und nicht-argumentativ, wurden die drei Klassen bezüglich der argumentativen Klassifikation, zu einer Klasse zusammengefasst. So gab es in den ursprünglichen 90 Essays insgesamt 90 'Major Claim's, 429 'Claim's und 1033 'Premise's, die zu 1552 Segmenten mit

[4] `https://www.freebase.com/`

[5] `https://docs.google.com/file/d/0B7XkCwpI5KDYaDBDQm1tZGNDRHc`

der Bezeichnung 'ARGUMENTATIVE' umbenannt wurden. Im dritten Experiment hingegen wurden nur die 'Major Claim's und 'Claim's als 519 Einheiten mit der Bezeichnung 'Claim' zusammengefasst und umbenannt, um die Relationen zwischen Claim und Premise besser klassifizieren zu können, ohne 'Major Claim' als unnötiges und zusätzliche Klassifizierung. Die Einstufung eines argumentativen Segments als 'Major Claim', anstatt nur als 'Claim', wurde von Stab und Gurevych eingeführt, aber nur in wenigen Forschungen bisher weiter verwendet. In meiner Arbeit wird deshalb auch nur zwischen 'Claim' und 'Premise' unterschieden und 'Major Claim' als Annotierung nicht verwendet.
Da nicht nur die Ground Truth für die Evaluation im ersten Experiment segmentiert sein muss, sondern auch der nicht argumentative Teil des Textes, wurde als zweiter Datensatz die Segmentierung von 'Approaches to Automatic Argumentative Unit Segmentation' von Johannes Kiesel verwendet (ab hier 'Text Segments' genannt), die über 95% aller inneren Satzgrenzen für die Segmente erkannt hat, aber auch zusätzliche Satzgrenzen hinzugefügt hat (Kiesel [2016]). Damit für das zweite und dritte Experiment jedes Segment klar als argumentativ und nicht argumentativ vorannotiert werden kann, habe ich die gegebenen Segmentgrenzen von 'Text Segments' genommen und sie mit der 'Ground Truth' abgeglichen, um gegebenenfalls neue Grenzen einzufügen, wo ein argumentatives Segment nicht klar von einem nicht argumentativen Segment getrennt war. Bei diesem Vorgang wurden zu den bisherigen 6798 Segmenten in den 90 Essays nur weitere 171 hinzugefügt, sodass es insgesamt 6969 Segmente gibt, die entweder als argumentativ oder nicht argumentativ annotiert sind.
Für das dritte Experiment bezüglich Relationenfindung zwischen Claim und Premise, musste jedes argumentative Segment mit jedem anderen argumentativen Segment kombiniert und auf Relationen annotiert werden (also ob zwischen zwei Segmenten ein 'attack' oder 'support' vorliegt, nach der Annotation von Stab und Gurevych). Bei dieser Umstrukturierung der Daten entstanden 14873 Relationen, wovon 1473 als true annotiert sind, also eine Beziehung haben.

AMC Bei AMC, Kurzform für Argumentaion Mining Corpus, handelt es sich um einen Korpus, bereitgestellt von Lucas Carstens[6] von dem Imperial College London. Dieser Korpus hat insgesamt 2274 Textaussagen zu 'UKIP', welches für 'United Kingdom Independence Party' steht und eine politische Partei in England ist. Das Thema ist deshalb überwiegend aus dem Bereich der Politik und wurde aus dem Web von verschiedenen News Seiten zusammengestellt. Jedes Beispiel hat dabei eine Zwei-Satz-Struktur, wobei der erste Satz eine

[6] `http://www.doc.ic.ac.uk/~lc1310/corpusCategories.xlsx`

Aussage enthält (also der Claim), während der zweite Satz ausschlaggebend für die Annotierung ist, indem er positiv oder negativ mit Begründung darauf reagiert (also die Premise). Insgesamt wurden 1394 Beispiele mit 'neutral', 419 mit 'attack' und 461 mit 'support' annotiert. 'neutral' bedeutet hierbei, dass die Sätze in keiner Verbindung stehen und damit unabhängige Aussagen beinhalten. 'attack' bedeutet, dass im zweiten Satz versucht wird die Aussage im ersten Satz zu widerlegen und 'support', dass im zweiten Satz die Aussage aus dem ersten Satz unterstützt wird, durch z.B. eine bekräftigende Aussage. In dem Korpus liegen noch weitere Themen neben UKIP vor, wie 'Movies', 'Evolution' oder 'Religion', haben aber als Quelle nur Kommentare und Foreneinträge. Die Zwei-Satz-Struktur wurde hier verworfen und beide Teile beinhalten oft nur unvollständige Sätze mit Schreibfehlern, bei denen oft auch die Annotierung in 'neutral', 'attack' und 'support' nicht nachvollziehbar ist. Da das Prinzip von Claim und Premise aus diesen Daten nicht mehr ersichtlich ist und damit nach der Definition keine argumentative Struktur bilden, wurden diese für meine Experimente nicht verwendet.
In der dazugehörigen Arbeit 'Towards relation based Argumentation Mining' (Carstens and Toni [2015]) von 2015 konnte mit Hilfe von Bag-of-Words, Relational und Sentential Features eine Klassifikationsgenauigkeit von 77,75% erzielt werden. Für meine Experimente wurden die support- und attack-Beispiele als argumentativ zusammengefasst und die neutral-Beispiele als nicht argumentativ umbenannt.

Kapitel 5

Experimentbeschreibung und Resultate

Dieses Kapitel beschreibt die einzelnen Experimente und zeigt ihre Ergebnisse, mit deren Hilfe die Thesen aufgezeigt werden sollen. Die drei Experimente spiegeln dabei die ersten drei Thesen wieder, ob man mit Worteinbettung die Textsegmentierung verbessern (Kapitel 5.1), Segmente als argumentativ bestimmen (Kapitel 5.2) und Relationen zwischen Segmenten erkennen kann (Kapitel 5.3). Alle Experimente wurden dabei so durchgeführt, dass die Klassifikationsdaten balancierte Klassen beinhalteten, d.h. dass 50% der Daten als argumentativ annotiert waren und die anderen 50% als nicht argumentativ. Weiterhin wurden die Resultate vom Bag-of-Words Verfahren als Vergleichswerte verwendet. Den Thesen vier bis sechs wurden nachgegangen, indem man zum einen für alle Experimente die vier verschiedenen Word2Vec-Modelle uppsala, text8, wiki14 und news verwendet hat, für das maschinelle Lernen die Verfahren J48 und OneR verwendet wurden und man beim zweiten Experiment bestimmte Bestandteile aus den Segmenten und Sätzen der Klassifikationsdaten extrahiert hat. Sollte der Extremfall entstehen, dass durch ein Modell über 10% der Beispiele in den Klassifikationsdaten kein Wert berechnet werden konnte, weil der Wortschatz einfach nicht groß genug war, dann wird hier das Endergebnis verworfen, weil man alle nicht evaluierten Beispiele automatisch als falsch klassifiziert einstufen müsste. Dies wäre nicht mehr repräsentativ und wird deshalb als Misserfolg gewertet.

5.1 Experiment 1: Verbesserung der Textsegmentierung

Mit diesem Experiment soll der ersten These 'Durch Worteinbettungen und Kosinusähnlichkeit kann man erkennen, ob benachbarte Textsegmente zusammen eine argumentative Aussage bilden.' nachgegangen werden. Die Problemstellung zu diesen Experiment wurde bereits in Kapitel 2.1.1 'Textsegmentierung' erläutert.

5.1.1 Ablauf

Hier wird so vorgegangen, dass die Grenzwörter eines jeden Segments genommen werden und mit den benachbarten Grenzwörtern der anderen Segmente zu einem Kosinusähnlichkeitswert verrechnet werden. Dieser berechnete Wert soll Aufschluss darüber geben, ob die Segmente eine argumentative Aussage bilden. Da durch fehlerhafte Segmentierung ein argumentatives Segment nach der Ground Truth aufgetrennt wurde, wird so durch Worteinbettung versucht die Relationen zwischen diesen beiden Segmenten zu erkennen und wieder zusammenzuführen.

5.1.2 Resultate

In Tabelle 5.1 ist zu sehen, welche Genauigkeit erreicht werden konnte, indem man durch Worteinbettung die Grenzworte nimmt und daran beurteilt, ob die Segmente zusammengehören, weil sie eine argumentative Aussage bilden. Man sieht, dass die Werte mit maximal 57.68% nicht sehr ausschlaggebend sind und mit OneR erzielt wurden. Das news-Modell konnte hierbei keine sinnvollen Ergebnisse erzielen, da bereits bei der Berechnung über 50% der Daten nicht evaluiert werden konnten, weil die Wörter nicht im Vokabular des Modells waren. Das liegt daran, dass an den Grenzen oft Stoppwörter stehen und diese im news Korpus bereits rausgefiltert wurden. Weiterhin sei erwähnt, dass durch den simpleren Bag-of-Words Algorithmus in Weka ein Wert von 69.06% erzielt wurde.

5.2 Experiment 2: Argumentativitäts-Klassifikation

In diesem Experiment soll These 2 nachgegangen werden, ob die Wortvektoren, die mit Word2Vec erstellt wurden, vorhersagen können, ob ein Segment argumentativ ist oder nicht. Da hierzu keine Relationen zwischen Segmenten

	Trainingsdaten			
Tasks	uppsala	text8	wiki14	news
Border Relations	56.61%	57.68%	55.38%	-

Tabelle 5.1: Erzielte Genauigkeiten bei der Grenzrelationen-Klassifikation. J48 hat in allen Fällen schlechtere Ergebnisse erzielt als OneR. Durch zufällige Klassifikation wird eine Genauigkeit von 50% erreicht.

berechnet werden, wird als Feature auch nicht die Kosinusähnlichkeit verwendet, wie in den anderen Experimenten, sondern die Wortvektoren selber.

5.2.1 Ablauf

Um einen mehrdimensionalen Vektor für ein Segment zu erhalten, wurden aus allen Wortvektoren in einem Segment ein Durchschnittsvektor erstellt. Lagen z.B. in einem Segment zwei Wörter mit den Wortvektoren w1 = (-1.0,0.0,1.0) und w2 = (0.0,0.5,1.0) vor (natürlich in viel höheren Dimensionen), dann wurde daraus der Durchschnittsvektor Ø= (-0.5,0.25,1.0) berechnet. Die Wortvektoren haben je nach Modell und Trainingskorpus eine Dimension von 100, 300 oder 1000 Werten, die damit die Features darstellen. Das Klassifizierungsverfahren J48 lernt in Weka von der Trainingsmenge eine Funktion, um allen Wortvektoren eine Klasse zuzuordnen. Diese Zuordnung soll möglichst der tatsächlichen Zuordnung nach der Ground Truth entsprechen.
Neben den Bag-of-Words Verfahren, wurden zum Vergleich aus dem AAEC-Korpus noch Positionsmerkmale als Structural Features entnommen. Die Motivation, diese Features auch noch zu extrahieren, lag darin, dass man schon an der Ground Truth erkennen konnte, dass z.B. 'Claim's sich mehr am Anfang eines Textes bzw. Paragraphen aufhalten. Dagegen hielten sich mit 'Premise' annotierte Segmente eher am Ende dieser Struktur auf. Um diesen Eigenschaften genauer nachzugehen, wurden neben den Durchschnittsvektoren noch 3 weitere Features 'Sentence Location', 'Paragraph Location' und 'Essay Location' hinzugefügt. Diese können die Werte 'start', 'mid' und 'end' annehmen, je nachdem wo sich das Segment im Satz, im Paragraphen oder im gesamten Aufsatz befindet. Sentence Location hat dazu den Wert 'complete', wenn es sich bei dem Segment um den ganzen Satz handelt, und Paragraph Location und Essay Location den zusätzlichen Wert 'title', wenn es sich um die Überschrift handelt, wo sich das Segment befindet.

5.2.2 Resultate

In Tabelle 5.2 sind die Resultate zum zweiten Experiment zu sehen, bei der es um die Klassifizierung der Segmente in argumentativ und nicht argumentativ ging, mit Hilfe von Wortvektoren und Positionsmerkmalen.

	Trainingsdaten			
Features	uppsala	text8	wiki14	news
Wortvektoren	69.59%	69.97%	71.01%	69.09%
Wortvektoren + Position	80.33%	80.68%	81.02%	81.13%

Tabelle 5.2: Erzielte Genauigkeit bei der Klassifizierung in argumentativ und nicht argumentativ. Die Klassifizierung nur anhand von Positionsmerkmalen hat eine Genauigkeit von 78.55% erzielt und das Bag-of-Words Verfahren eine Genauigkeit von 71.06%. Durch zufällige Klassifikation wird eine Genauigkeit von 50% erreicht.

Mit den Wortvektoren, die für die jeweiligen Segmente im AAEC Korpus zu einem Durchschnittsvektor berechnet wurden, konnte eine Genauigkeit von 71.01% erzielt werden. Die Klassifizierung allein anhand von Positionsmerkmalen hat aber bereits einen Wert von 78.55% erzielt. Nimmt man beide Features zusammen konnte ein Maximalwert von 81.02% erreicht werden. Das Bag-of-Words Verfahren hat in diesem Experiment einen Wert von 71.06% erzielt und ist damit knapp besser als der vom wiki14-Modell erreichte Wert.
Interessant ist es noch zu betrachten, welche Werte bei den einzelnen Positionsmerkmalen erzielt werden konnten. Dies wird in Abbildung 5.1 gezeigt.

Zu sehen ist, dass mit 82.62% die meisten Argumente mit hoher Wahrscheinlichkeit am Ende eines Satzes stehen. Weiterhin stehen mit wachsender Wahrscheinlichkeit Argumente am Ende eines Paragraphen mit 66.86% und 73.59% und der Großteil der argumentativen Aussagen sind in der Mitte und am Ende eines Essays zu finden mit 74.81% und 60.73%. Der Titel selber wurde immer als nicht argumentativ annotiert, obwohl dieser oft dem 'Major Claim' entspricht. Dies ist wohl eine Konvention von Stab und Gurevych, um keine Redundanz in der Annotation zu haben (der Titel zählt selber nicht als argumentativer Bestandteil). Allgemein lässt sich noch sagen, dass Sentence Location und Essay Location mit 71.36% und 73.55% die größte Aussagekraft beinhalten.

```
Sentence Location:
    start     -> UNARGUMENTATIVE   (1057/1491)     -> 70.89%
    mid       -> ARGUMENTATIVE     (2593/3798)     -> 68.27%
    end       -> ARGUMENTATIVE     (1232/1491)     -> 82.62%
    complete -> ARGUMENTATIVE      (91/189)        -> 48.15%
(4973/6969 instances correct -> 71.36%)

Paragraph Location:
    title     -> UNARGUMENTATIVE   (125/125)       -> 100.00%
    start     -> UNARGUMENTATIVE   (868/1639)      -> 52.96%
    mid       -> ARGUMENTATIVE     (2498/3736)     -> 66.86%
    end       -> ARGUMENTATIVE     (1081/1469)     -> 73.59%
(4572/6969 instances correct -> 65.60%)

Essay Location:
    title     -> UNARGUMENTATIVE   (125/125)       -> 100.00%
    start     -> UNARGUMENTATIVE   (940/1229)      -> 76.48%
    mid       -> ARGUMENTATIVE     (3458/4622)     -> 74.81%
    end       -> ARGUMENTATIVE     (603/993)       -> 60.73%
(5126/6969 instances correct -> 73.55%)
```

Abbildung 5.1: Erzielte Genauigkeiten in der Klassifikation von Segmenten in argumentativ und nicht argumentativ anhand von den Positionsmerkmalen 'sentence location', 'paragraph location' und 'essay location' einzeln klassifiziert.

5.3 Experiment 3: Relationenfindung

Im letzten und größten Experiment steht die These 3 im Vordergrund, ob man mit Word2Vec und Kosinusähnlichkeit Relationen zwischen argumentativen Segmenten erkennen kann. Weiterhin wurde auch These 5 genauer nachgegangen, bei der es darum ging, ob man durch das Extrahieren bestimmter Bestandteile aus Segmenten und Sätzen bessere Kosinusähnlichkeitswerte berechnen lassen.

5.3.1 Ablauf

Für These 5 wurden die Segmente aus dem AAEC und die Sätze aus dem AAEC Klassifizierungskorpus genommen und bestimmte Bestandteile extrahiert, sodass eventuell bessere Kosinusähnlichkeitswerte berechnet werden können. Dies ist sinnvoll, weil Wörter die 'Noise' (also Rauschen) verursachen nicht mehr in den Daten stecken. Die Extrahierungen wurden mit den Stanford CoreNLP Tools durchgeführt und werden im Folgenden beschrieben.

Naive Bei diesem naiven Ansatz soll zunächst aus allen Wörtern eines Segments mit allen Wörtern aus einem anderen Segment die Kosinusähnlichkeit berechnet werden, indem man vorher aus den jeweiligen Wortvektoren Durchschnittsvektoren berechnet. Sollte ein großer Wert hier entstehen, so müsste dies Aufschluss darüber geben, dass eine Relation zwischen zwei Segmenten vorliegt.

Nouns Bei dieser Filterung werden die Nomen aus dem Satz extrahiert. Analog zu 'Naive' werden dann alle Nomen eines Segments zu Durchschnittsvektoren berechnet, damit man danach die Kosinusähnlichkeit zu anderen Segmenten bzw. Sätzen berechnen kann.

Nouns Max Ähnlich wie bei der vorherigen Filterung, werden zunächst alle Nomen aus den Segmenten extrahiert. Der Unterschied besteht nun darin, dass die Kosinusähnlichkeit von jedem Nomen eines Segments in Kombination mit jedem Nomen eines anderen Segments berechnet wird. Der höchste Wert, der hierbei entsteht, wird als Ergebnis übernommen. Der Ausnahmefall, dass die Kosinusähnlichkeit von zwei gleichen Nomen berechnet wird (was als Ergebnis 1 zur Folge hätte), wird hierbei ausgeschlossen und nur alle anderen Kombinationen werden betrachtet.

Noun Phrases Bei dieser Filterung werden die Nominalphrasen aus dem Satz extrahiert. Dies ist meist eine Kombination aus Nomen mit Artikel oder

Adjektiv. Analog zu 'Naive' und 'Nouns' wird dann hier auch die Kosinusähnlichkeit aus den Durchschnittsvektoren berechnet.

Noun Phrases Max Analog zu Nouns Max, wird auch hier nur der maximale Wert aus der Kombination aus allen Noun Phrases aus beiden Segmenten gewählt und der Ausnahmefall, dass zwei Noun Phrases gleich sind, in allen Kombinationen ausgeschlossen.

Um Forschungsfrage 6 nachzugehen, ob man einen Grenzwert festlegen kann in den Kosinusähnlichkeitswerten ab dem ein Segment als argumentativ gilt oder nicht, wurden alle Experimente in 10-Fold Cross-Validation einmal mit dem Klassifizierungsverfahren OneR und einmal mit J48 durchgeführt. Das Verfahren OneR versucht mehrere Bereiche in den Kosinusähnlichkeitswerten zwischen -1 und 1 aufzustellen und diese danach zu klassifizieren, dass sie die besten Ergebnisse erzielen. J48 versucht dies mit einem Grenzwert. Sollte J48 bessere Ergebnisse in den meisten Experimenten erzielt haben als OneR, so sollte dies Aufschluss darüber geben, dass man ab einem bestimmten Grenzwert in den Kosinusähnlichkeitswerten beurteilen kann, wann ein Segment als argumentativ gilt oder nicht. Eine ausführliche Erklärung zu OneR und J48 kann im Abschnitt 3.3 'Maschinelles Lernverfahren mit Weka' nachgelesen werden.

Weiterhin wurde in zusätzlichen Experimenten die Lemmatisierung und die Koreferenz-Auflösung auf den Datensätzen angewandt, um die Ergebnisse positiv zu verbessern. Bei der Lemmatisierung werden alle Wörter auf ihre einfache Form ohne Fallanpassung zurückgesetzt und in der Koreferenz-Auflösung werden Referenzen auf ihr Bezugswort aufgelöst. Eine ausführliche Erklärung zu diesen Vorgängen kann im Abschnitt 3.1 'Datenextrahierung mit den Stanford CoreNLP Tools' nachgelesen werden.

5.3.2 Resultate

Im Folgenden sollen die Ergebnisse des zweiten Experiments aufgezeigt werden: einmal ohne weiteres Preprocessing, einmal mit Lemmatisierung und einmal mit Koreferenzauflösung. Dabei wurden, wie in den vorherigen Experimenten auch, die vier Modelle uppsala, text8, wiki14 und news verwendet, aber dazu wurden noch bestimmte Bestandteile aus den Klassifikationsdaten extrahiert, wie im vorherigen Abschnitt 5.3.1 'Ablauf' geschildert.

Normalen Ergebnisse ohne Preprocessing

In diesem Abschnitt sollen die Ergebnisse gezeigt werden, ohne zusätzliches Preprocessing. Alle Ergebnisse bezüglich der AAEC Klassifikationsdaten können in der Tabelle 5.3 eingesehen werden.

	Trainingsdaten			
Filter	uppsala	text8	wiki14	news
naive	54.41%	53.63%	52.65%	53.33%
nouns	57.86%	57.65%	57.63%	58.77%
nouns-max	51.94%*	53.98%*	52.21%	56.04%
noun-phrases	56.37%	57.06%	55.89%	53.02%
noun-phrases-max	50.34%	55.47%	54.24%	53.52%

Tabelle 5.3: Erzielte Genauigkeiten zum Experiment 'Relationenfindung mit Word2Vec als semantisches Feature' mit den Klassifikationsdaten von AAEC. Das Bag-of-Words Verfahren hat eine Genauigkeit von 58.76% erreicht. Die besten Ergebnisse stammen in den meisten Fällen von J48. Die besten Ergebnisse die von OneR stammen sind mit einem * markiert. Durch zufällige Klassifikation wird eine Genauigkeit von 50% erreicht.

Man kann sehen, dass bei der nouns-Filterung die besten Ergebnisse erzielt wurden, aber kein Wert über 58.77% gelangt und damit auch nicht sehr aussagekräftig ist. Anzumerken sei hier noch, dass das Bag-of-Words Verfahren hier ein Ergebnis von 58.76% erreicht hat und damit nur leicht besser ist als die meisten Ergebnisse. Weiterhin waren die besten Ergebnisse in 17 Fällen von J48, in 2 Fällen von OneR und einmal annähernd gleich. In Tabelle 5.4 können die Ergebnisse der AMC Klassifikationsdaten eingesehen werden.

Aus diesen Resultaten ist nun ersichtlich, dass das Modell von wiki14 mit über 60% die mit Abstand besten Resultate erzielt hat. Anzumerken ist hier, dass das Bag-of-Words Verfahren hier ein Ergebnis von 66.08% erreicht hat, welches wiki14 mit den vorgefilterten Nomen um fast 2.63% geschlagen hat. Weiterhin waren die besten Ergebnisse in 12 Fällen von J48, in 3 Fällen von OneR und 5 mal annähernd gleich.

Ergebnisse mit entfernter Koreferenz

In diesem Abschnitt werden die Ergebnisse gezeigt, bei denen aus den Klassifikationsdaten die Koreferenz aufgelöst wurde. Alle Ergebnisse bezüglich der AAEC Klassifikationsdaten können in der Tabelle 5.5 eingesehen werden.

Zu sehen ist, dass fast alle Werte sich leicht verbessert haben im Durchschnitt von 0.61%. Dabei haben sich die Resultate von OneR im Durchschnitt

	Trainingsdaten			
Filter	uppsala	text8	wiki14	news
naive	59.77%	59.77%	63.58%	53.18%
nouns	57.72%	56.89%	68.71%	60.02%
nouns-max	59.10%*	55.24%*	54.89%*	57.86%
noun-phrases	57.72%	56.89%	64.89%	54.63%
noun-phrases-max	53.83%	55.14%	59.03%	54.62%

Tabelle 5.4: Erzielte Genauigkeiten zum Experiment 'Relationenfindung mit Word2Vec als semantisches Feature' mit den Klassifikationsdaten von AMC. Das Bag-of-Words Verfahren hat eine Genauigkeit von 66.08% erreicht. Die besten Ergebnisse stammen in den meisten Fällen von J48. Die besten Ergebnisse die von OneR stammen sind mit einem * markiert. Durch zufällige Klassifikation wird eine Genauigkeit von 50% erreicht.

	Trainingsdaten			
Filter	uppsala	text8	wiki14	news
naive	54.85%	54.65%	55.50%	54.38%
nouns	58.51%	58.33%	59.31%	57.23%
nouns-max	50.76%	54.46%	51.48%	55.37%
noun-phrases	55.69%	58.19%	56.61%	56.88%
noun-phrases-max	53.03%	55.48%	53.00%	54.02%

Tabelle 5.5: Erzielte Genauigkeiten zum Experiment 'Relationenfindung mit Word2Vec als semantisches Feature' mit den Klassifikationsdaten von AAEC und Koreferenzauflösung. Das Bag-of-Words Verfahren hat eine Genauigkeit von 57.13% erreicht. Die besten Ergebnisse stammen in den meisten Fällen von J48. Durch zufällige Klassifikation wird eine Genauigkeit von 50% erreicht.

um 0.49% und von J48 um 0.72% verbessert. Anzumerken sei hier noch, dass das Bag-of-Words Verfahren ein Ergebnis von 57.13% erreicht hat und das Modell wiki14 mit extrahierten Nomen dies um 2.18% geschlagen hat. Weiterhin waren die besten Ergebnisse in 16 Fällen von J48, in keinem Fall von OneR und 4 mal annähernd gleich. Von insgesamt 40 Werten haben sich 27 positiv verändert. In Tabelle 5.4 können die Ergebnisse der AMC Klassifikationsdaten eingesehen werden.

In diesem Experiment haben sich nur 17 von 40 Werten leicht verbessert, alle anderen verschlechtert. Damit sind die Ergebnisse im Durchschnitt um 0.35% gefallen, wobei sich die Resultate von OneR im Durchschnitt um 0.37% und von J48 um 0.32% verschlechtert haben. Die besten Ergebnisse waren in 13

	Trainingsdaten			
Filter	uppsala	text8	wiki14	news
naive	60.80%	60.23%	60.74%	56.36%
nouns	58.57%	56.03%	67.98%	59.33%
nouns-max	59.03%	58.73%	62.78%	53.25%*
noun-phrases	53.03%	55.48%	59.20%	53.42%*
noun-phrases-max	58.63%*	55.01%*	54.27%*	59.39%*

Tabelle 5.6: Erzielte Genauigkeiten zum Experiment 'Relationenfindung mit Word2Vec als semantisches Feature' mit den Klassifikationsdaten von AMC und Koreferenzauflösung. Das Bag-of-Words Verfahren hat eine Genauigkeit von 66.88% erreicht. Die besten Ergebnisse stammen in den meisten Fällen von J48. Die besten Ergebnisse die von OneR stammen sind mit einem * markiert. Durch zufällige Klassifikation wird eine Genauigkeit von 50% erreicht.

Fällen von J48, in 6 Fällen von OneR und 1 mal annähernd gleich. Weiterhin konnte das Ergebnis vom Bag-of-Words Verfahren mit 66.88% nicht geschlagen werden.

Ergebnisse mit Lemmatisierung

In diesem Abschnitt werden die Ergebnisse gezeigt, wenn man sowohl auf den Klassifikationsdaten als auch auf den Trainingsdaten Lemmatisierung durchführt. Alle Ergebnisse bezüglich der AAEC Klassifikationsdaten können in der Tabelle 5.7 eingesehen werden.

	Trainingsdaten			
Filter	uppsala	text8	wiki14	news
naive	54.65%	54.72%	51.02%	53.93%
nouns	58.57%	56.84%	57.00%	56.76%
nouns-max	53.07%*	53.72%	51.56%	54.69%
noun-phrases	55.21%	56.72%	52.67%	52.81%
noun-phrases-max	51.26%	53.52%	52.39%	53.66%

Tabelle 5.7: Erzielte Genauigkeiten zum Experiment 'Relationenfindung mit Word2Vec als semantisches Feature' mit den Klassifikationsdaten von AAEC und Lemmatisierung. Das Bag-of-Words Verfahren hat eine Genauigkeit von 58.76% erreicht. Die besten Ergebnisse stammen in den meisten Fällen von J48. Die besten Ergebnisse die von OneR stammen sind mit einem * markiert. Durch zufällige Klassifikation wird eine Genauigkeit von 50% erreicht.

In diesem Versuch haben sich die Werte im Durchschnitt um 0.47% verschlechtert. Dabei haben sich die Resultate von OneR im Durchschnitt um 0.37% und die von J48 um 0.59% verschlechtert. Weiterhin waren die besten Ergebnisse in 14 Fällen von J48, in einem Fall von OneR und 5 mal annähernd gleich gewesen. Von insgesamt 40 Werten haben sich nur 13 leicht verbessert, alle restlichen überwiegend verschlechtert und das Ergebnis des Bag-of-Words Verfahrens von 58.76% konnte nicht geschlagen werden. In Tabelle 5.8 können die Ergebnisse der AMC Klassifikationsdaten eingesehen werden.

	Trainingsdaten			
Filter	uppsala	text8	wiki14	news
naive	60.45%	63.30%	63.64%	55.51%*
nouns	58.06%	68.30%	68.07%	59.84%
nouns-max	60.18%	63.01%*	63.35%*	58.02%*
noun-phrases	56.82%	60.23%	59.61%	51.89%
noun-phrases-max	56.58%	61.19%	62.16%*	54.62%*

Tabelle 5.8: Erzielte Genauigkeiten zum Experiment 'Relationenfindung mit Word2Vec als semantisches Feature' mit den Klassifikationsdaten von AMC und Lemmatisierung. Das Bag-of-Words Verfahren hat eine Genauigkeit von 65.28% erreicht. Die besten Ergebnisse stammen in den meisten Fällen von J48. Die besten Ergebnisse die von OneR stammen sind mit einem * markiert. Durch zufällige Klassifikation wird eine Genauigkeit von 50% erreicht.

In diesem Experiment haben sich 25 von 40 Werten verbessert. Im Durchschnitt sind die Werte in Gegensatz zu den AAEC Klassifikationsdaten oder den AMC Koreferenz Experimenten dieses mal um 1.53% gestiegen. Die Resultate von OneR sind dabei um 1.77% und von J48 um 1.31% gestiegen. Die besten Ergebnisse waren in 11 Fällen von J48, in 5 Fällen von OneR und 4 mal annähernd gleich. Besonders anzumerken sei, dass die Werte vom text8-Modell sich im Vergleich zu den Ausgangsdaten im Durchschnitt um 5% verbessert haben (Vergleich Tabelle 5.4). Das Bag-of-Words Verfahren hat eine Genauigkeit von 65.28% und konnte mit der Nomen-Extrahierung sowohl vom text8-Modell als auch dem wiki14-Modell um ungefähr 3% geschlagen werden.

Kapitel 6

Diskussion

In diesem Kapitel soll darüber diskutiert werden, ob die Thesen, die in der Einleitung dieser Arbeit genannt wurden, bestätigt werden konnten. Dabei wird zuerst auf die Ergebnisse der Experimente eingegangen und danach auf mögliche Fehlerquellen aus Sicht der Trainingsdaten von Word2Vec, den Klassifikationsdaten und dem Framework Word2Vec selber.

6.1 Experimentresultate

In diesem Abschnitt soll es darum gehen, ob die Thesen, die in der Einleitung formuliert wurden, durch die Experimente erfüllt wurden. Im Kapitel 5 'Experimentbeschreibung und Resultate' wurde auf eine Sache nicht eingegangen, nämlich die Daten, zu denen keine Kosinusähnlichkeit berechnet werden konnte, weil die Wörter der Sätze bzw. Segmente nicht im Vokabular des Modells zu finden waren. Diese 'Ausschussrate' bezüglich der verschiedenen Ansätze und Modelle soll in folgender Tabelle 6.1 aufgezeigt werden.

Wie bereits in den Resultaten des ersten Experiments erwähnt, sind hier die meisten Daten weggefallen, da nur die Wörter am Rand von Segmenten betrachtet wurden und diese oftmals nur Stoppwörter waren. Das Modell, welches aus den Trainingsdaten von 'news' entstanden ist und keine Stoppwörter enthält, erzeugt hierbei eine sehr hohe Ausschussrate von über 66.06%, aber auch beim uppsala Modell gab es eine Ausschussrate von 9.01%. Besser hingegen sah es in Experiment 3 mit der Relationenfindung zwischen ganzen Segmenten oder Sätzen aus. Hier stieg die Ausschussrate nie über 2.79%.

	Trainingsdaten			
Filter auf Klassifikationsdaten	uppsala	text8	wiki14	news
Exp1: border-relation(AAEC)	09.01%	01.92%	00.38%	66.06%
Exp3: naive(AAEC)	00.00%	00.00%	00.00%	00.00%
Exp3: nouns(AAEC)	02.79%	02.78%	02.36%	02.38%
Exp3: noun-phrases(AAEC)	00.94%	01.10%	00.94%	00.94%
Exp3: naive(AMC)	00.00%	00.00%	00.00%	00.00%
Exp3: nouns(AMC)	00.40%	00.13%	00.04%	00.26%
Exp3: noun-phrases(AMC)	00.44%	00.40%	00.00%	00.40%

Tabelle 6.1: Prozentsatz an Daten in den Experimenten mit Word2Vec, zu denen mit einem Modell keine Kosinusähnlichkeit berechnet werden konnte, weil das Vokabular zu klein war.

6.1.1 Experiment 1: Verbesserung der Textsegmentierung

Mit Experiment 1 sollte die **1. These** 'Durch Worteinbettungen und Kosinusähnlichkeit kann man erkennen, ob benachbarte Textsegmente zusammen eine argumentative Aussage bilden.' bewiesen werden. Die Ergebnisse hingegen zeigen kein ausschlagendes Ergebnis und wurden auch ausschließlich mit OneR erzielt, was auch die **6. These** 'Ab einem bestimmten Kosinusähnlichkeitswert kann man entscheiden, ob eine Relation zwischen Segmenten bzw. Sätzen vorliegt oder nicht.' nicht unterstützt. Besonders ernüchternd wirken die Ergebnisse, wenn man sie mit dem Ergebnis des simpleren Bag-of-Words Algorithmus vergleicht. Dieser hat einen Wert von 69,09% erreicht.
Als Hauptgrund für die schlechten Ergebnisse mit Word2Vec seien hier die Stoppwörter zu erwähnen. Wie bereits in 'Measuring Word Significance using Distributed Representations of Words' (Schakel and Wilson [2015]) erwähnt (Kapitel 3.2.2), kann Word2Vec nur dann sinnvolle Wortvektoren bilden, wenn ein Wort nur in einem gewissen Maß oft verwendet wird, um es zu umschreiben. Wörter, die allerdings in einem gesamten Korpus in einer hohen Frequenz auftreten, bilden auch keine sinnvollen Vektoren zu anderen Wörtern. Da in diesem Experiment aber scheinbar 2/3 der Daten zur Kosinusähnlichkeitsberechnung aus Stoppwörtern bestand (siehe Ausschussrate von news Modell im ersten Experiment), konnten auch keine guten Werte berechnet werden, sondern sind in den meisten Fällen zufällig.

6.1.2 Experiment 2: Argumentativitäts-Klassifikation

In Experiment 2 sollte die **2. These** bewiesen werden, dass man mit Wortvektoren Segmente in argumentativ und nicht-argumentativ klassifizieren kann. Mit 71.01% ist hier ein guter Wert entstanden, der aber auch nur knapp an das Resultat von Bag-of-Words mit 71.06% heran reicht. Die Positionsmerkmale als zusätzliche Features haben aber gezeigt, dass durchaus höhere Werte möglich sind, auch wenn diese zu den Structural Features zählen, und nicht den Contextual Features, wie die semantischen Eigenschaften durch Worteinbettung. In Kombination wurde ein Maximalwert von 81.02% erreicht. Da das Bag-of-Words Resultat allein durch aufgestellte Wortvektoren nicht geschlagen werden konnte, gilt auch dieser Forschungsansatz als nicht bewiesen.

6.1.3 Experiment 3: Relationenfindung

Im dritten und größten Experiment sollte der **3. These** nachgegangen werden 'Durch Worteinbettungen und Kosinusähnlichkeit als kontextuelle Features kann man Relationen zwischen 'Claim' und 'Premise' in argumentativen Texten erkennen.'. Dabei sind großteils nüchterne Ergebnisse entstanden. Allgemein lässt sich erst einmal sagen, dass mit den AMC Klassifikationsdaten bessere Ergebnisse erzielen ließen, weil hier ganze Sätze mit mehr Informationen miteinander verglichen wurden. Dies zeigt sich auch in den max-Ansätzen, bei denen immer nur der stärkste Kosinuswert gewählt wurde, anstatt alle Wörter in die Berechnung mit einzubeziehen. Diese fallen zu ihren normalen Konterpart, also nouns-max zu nouns und noun-phrases-max zu noun-phrases, immer schlechter aus. Weiterhin konnte man erkennen, dass durch Lemmatisierung bei dem text8-Modell die Ergebnisse stark verbessert wurden. Dagegen wurden durch Auflösung der Koreferenz die Ergebnisse der AAEC Klassifikationsdaten leicht besser. Trotz der großteils nüchternen Ergebnisse, die oftmals nicht einmal an die Vergleichswerte von Bag-of-Words heranreichen, haben sich einige Forschungsansätze bewahrheitet, welche nun nacheinander durchgegangen werden.

Bei **These 3** ging es darum, ob man mit Worteinbettung als kontextuelles Feature Relationen zwischen 'Claim' und 'Premise' erkennen kann. Diese Nachforschungen haben sich nicht wirklich bewahrheitet, da nur in wenigen Einzelfällen der Bag-of-Words Algorithmus geschlagen werden konnte. In nur drei Fällen hatte das wiki14-Modell mit extrahierten Nomen in den Klassifikationsdaten eine höhere Genauigkeit um 2 bis 3% als das Bag-of-Words Verfahren.

Anders hingegen sieht es mit der **6. These** aus 'Ab einem bestimmten Ko-

sinusähnlichkeitswert kann man entscheiden, ob eine Relation zwischen Segmenten bzw. Sätzen vorliegt oder nicht.'. Das Klassifizierungsverfahren J48, welches versucht ab einem bestimmten Wert zu klassifizieren, hat in 82 von 104 Fällen (die 16 Fälle bei denen beide Verfahren annähernd gleiche Ergebnisse hatten wurden nicht mit einbezogen) die besseren Ergebnisse hervorgebracht, anstatt OneR, welches versucht mehrere Bereiche in der Kosinusähnlichkeit zu diskretisieren, um die besten Ergebnisse zu erzielen. Neben dieser These konnte man aber weiterhin noch beobachten, dass OneR in den max-Ansätzen ('Nouns Max' und 'Noun Phrases Max') meist bessere Ergebnisse als J48 erzielt hatte.

These 4 konnte sich auch bewahrheiten. Das Modell wiki14, welches aus dem größten Trainingskorpus erstellt wurde und damit auch das größte Vokabular hat, hat in den meisten Experimenten die besten Ergebnisse erzielt. Dies ist besonders bei den AMC Klassifikationsdaten ersichtlich, da hier mit fast jeder Extrahierung bestimmter Bestandteile aus den Klassifikationsdaten eine Genauigkeit von über 60% erzielt wurde.

Die übrig gebliebene **These 5** konnte auch aufgezeigt werden. Durch das Filtern der Nouns aus den Segmenten bzw. Sätzen konnten fast immer die besten Ergebnisse erzielt werden. Da Nomen durch Word2Vec besser als Wortvektoren trainiert werden, als vergleichsweise Verben oder Adjektive, die in einer größeren Anzahl an Kontexten vorkommen, ist dieses Ergebnis auch nachvollziehbar. Etwas schlechtere, aber ähnliche Ergebnisse, konnten auch durch die Filterung der Noun Phrases erzielt werden. Das Hinzufügen von entsprechenden Adjektiven oder Artikeln als Begleitern zu Nomen, kann das Ergebnis sowohl verbessern als auch verschlechtern, je nachdem wie gut der Begleiter trainiert ist. Beispiele für solche Fälle sollen im nächsten Unterkapitel 'Mögliche Fehlerquellen' aufgezeigt werden.

6.2 Mögliche Fehlerquellen

In diesem Abschnitt soll es um die möglichen Fehlerquellen gehen, die die Ergebnisse negativ beeinflusst haben, und mögliche Alternativen, mit denen man die Ergebnisse verbessern könnte. Dabei werden drei Quellen betrachtet: Die Daten zum Trainieren der Modelle mit Word2Vec, die Testdaten zur Klassifizierung und das Framework Word2Vec selber, mit dem die Features erstellt wurden.

6.2.1 Die Trainingsdaten für die Word2Vec-Modelle

Als Trainingsdaten für die Word2Vec-Modelle wurden vier Korpora gewählt, mit unterschiedlichen Größen und Themen: Das kleine Trainingsdatenset **'uppsala'**, welches nur aus 1.489 Texten bestand, aber dafür alles argumentative Essays; der kleine Wikipediakorpus **'text8'**, welcher als Paradebeispiel für viele Experimente in der maschinellen Sprachverarbeitung verwendet wird; der große Wikipediakorpus **'wiki14'**, welcher über 1.219.502 Artikel von 2014 enthält und der große Google News Korpus **'news'**, welcher aus über 100 Milliarden Wörtern besteht ohne Stoppwörter. Obwohl diese Korpora von unterschiedlicher Quelle und Größe sind, konnte keiner in den Ergebnissen voll überzeugen und durchgängig ausschlaggebendere Ergebnisse als das Bag-of-Words Verfahren liefern.

Allgemein lässt sich sagen, dass jeder Korpus nur die Hälfte seiner Aufgabe erfüllt. Ein perfekter Trainingskorpus müsste groß genug sein, um zu jeden Wort sinnvolle Vektoren aufstellen zu können, und argumentativ, damit ein Wort Relationen zu anderen Wörtern aufbauen kann, die ebenfalls oft in dessen Zusammenhang genannt werden, um für etwas zu überzeugen. Uppsala ist ein argumentativer Korpus, aber mit nur geringen Wortschatz. Während text8, wiki14 und der news Korpora ein großes Vokabular haben, aber mit nur wenig oder gar keinen argumentativen Inhalten, da hier nur Definitionen oder Neuigkeiten erklärt werden.

Ein weiter Punkt ist der Aufbau der Trainingsdaten. Obwohl ein großes Vokabular gut ist, um alle Daten überhaupt erst einmal evaluieren zu können, so hat doch Word2Vec Schwierigkeiten Relationen zu den wichtigsten Wörtern aufzubauen, wenn diese zu weit auseinander stehen, also nicht in einem Fenster sind. Heutzutage sind die Wikipedia-Artikel meist zu weiträumig ausformuliert, ohne das Schlagwort selber, also den Titel, noch zu benutzen oder umschreiben diesen nur. Damit Word2Vec das Artikeltitelwort besser lernen kann, müsste man die Artikel so umformulieren, dass der Titel immer am Anfang eines jeden Abschnitts steht, gefolgt von den Schlagwörtern, die den Begriff in diesem Abschnitt beschreiben. Kurz gesagt bräuchte man einen Korpus, der supervised ist, bei dem also die wichtigsten Stellen zum Lernen schon selektiert sind. Die größte Schwierigkeit würde darin aber bestehen, die richtigen Schlagwörter aus jeden Abschnitt zu extrahieren.

Weiterhin kann, wie bereits in Kapitel 3.2.2 erwähnt, die Ähnlichkeitsverteilung einen negativen Einfluss auf die Berechnung der Kosinusähnlichkeit haben. Im Normalfall sollte die Ähnlichkeit von allen möglichen Wortpaaren

Modell	Durchschnittkosinusähnlichkeit
uppsala	0.1410
text8	0.0700
wiki14	0.0819
news	0.1305

Tabelle 6.2: Durchschnittkosinusähnlichkeit berechnet aus 10.000 zufälligen Wortpaaren auf den verschiedenen Modellen.

in einem Modell 0 ergeben. Durch Testreihen mit 10.000 zufällig ausgewählten Wortpaaren haben sich für die jeweiligen Modelle der verschiedenen Trainingsdaten aber abweichende Werte ergeben, siehe Tabelle 6.2. Der Kosinusähnlichkeitswert kann zwischen -1 und 1 liegen. Liegt der Durchschnittswert aber, wie hier zu sehen, bei uppsala oder news statt bei 0 eher bei 0.13 oder weiter entfernt, kann das die Ergebnisse, besonders im Vergleich zu anderen, leicht verfälschen. Die Korpora text8 und wiki14 haben bessere Werte, da diese auf großen Mengen von Trainingsdaten über 20 Iterationen trainiert wurden. Uppsala hingegen besteht nur aus einem kleinen Trainingsdatenset und news wurde scheinbar nur mit wenigen Iterationen trainiert (genauere Informationen, mit welchen Parametern der Google News Korpus trainiert wurde, sind nicht vorhanden).

Ein weiterer, aber meines Erachtens sehr wichtiger Punkt, ist die Frage, ob es überhaupt möglich ist neue Argumentationen vorherzusagen. Natürlich kann man aus alten Argumentationen Relationen lernen, wie dies auch mit dem Korpus uppsala gemacht wurde, aber lassen sich aus diesen 'alten' Relationen auf 'neue' Relationen schließen? Von Natur aus soll ein neuer argumentativer Text von neuen Themen überzeugen, z.B. ob es wichtig ist Kriegsflüchtlinge im eigenen Land aufzunehmen oder es wichtig wäre auf keine bestimmten Handelsabkommen einzugehen. Wenn nun aber alte Texte nie Stellungnahme zu diesen Themen genommen haben bzw. es aus der Sicht eines anderen Punktes argumentativ analysiert haben, dann ist es auch für ein Modell schwer diese neuen Begebenheiten richtig zu klassifizieren. Die Frage bleibt also bestehen: Ist es überhaupt möglich neue Argumentationen aus alten Texten richtig einzuschätzen? Schließlich will der Schreiber selber mit neuen, bisher nicht erwähnten, Argumenten überzeugen. Angeblich soll Word2Vec zwar auch von Menschen bisher unbekannte Relationen erkennen können, aber dass diese auch stimmen und auf neue Argumentationen passen sollen ist eher unwahrscheinlich.

Ein letzter Punkt, der sich aus dem vorherigen Absatz ebenfalls schließen lässt, ist die Tatsache, dass Argumentationen sich ständig ändern. So kann zum

Beispiel eine Sache, die bis dato als gut oder neutral klassifiziert wurde, plötzlich durch einen neuen Bericht als vollkommen negativ klassifiziert werden. Ein älteres Beispiel wäre das Thema 'Rauchen', welches bis 1962 ohne Sorge ausgeübt wurde. Doch dann erschien eine bahnbrechende epidemiologische Studie des Royal College of Physicians[1], welches die negativen Aspekte des Rauchens wissenschaftlich aufzeigte. Damit hatte sich auch das 'Modell' um das Wort 'Rauchen' in den folgenden Jahrzehnten stark geändert. Ein aktuelleres Beispiel ist das Thema 'Gluten'. Während vor noch kurzer Zeit Gluten nur ein Sammelbegriff war für Proteine aus bestimmten Getreidearten, so gilt es heute als 'schlechter' Bestandteil in vielen Lebensmitteln aufgrund von neuen wissenschaftlichen Studien.

6.2.2 Die Klassifikationsdaten

Ein großes Problem aus Sicht der Klassifikationsdaten ist die Vielzahl an Relationen, die es in Wirklichkeit gibt. Bereits 2013 wurde von Socher et al. in der Arbeit 'Reasoning With Neural Tensor Networks for Knowledge Base Completion' Socher et al. [2013] Experimente mit Relationen und neuronalen Netzen durchgeführt. Obwohl hier teilweise ausgezeichnete Ergebnisse mit bis zu 90.00% richtig klassifizierten Einheiten erzielt wurden, war der Versuchsaufbau ein anderer. Der Input für die Klassifikation bestand aus einer Dreiergruppe: einem Wort oder Wortgruppe A, einem Wort oder Wortgruppe B und einer Relationsphrase. Die Relationsphrase konnte zum Beispiel 'has part', 'similar to' oder 'has instance' sein. In den Ergebnissen konnte man erkennen, dass einige Relationen nur schwer zu erkennen waren. Bezogen auf meine Arbeit aber, gibt es theoretisch nur die Relationen 'supporting' oder 'attacking' in Bezug auf Argumentationen, die aber je nach Thema unterschiedlich ausfallen können und schwieriger zu klassifizieren sind als vergleichsweise 'part of'. Während es bei 'part of' eindeutige Lösungen gibt wie zum Beispiel 'Reifen sind ein Bestandteil vom Auto', so lassen sich je nach Argumentation 'supporting' und 'attacking' nicht eindeutig klassifizieren. Wie bereits im vorherigen Unterkapitel bezüglich Trainingsdaten für die Word2Vec-Modelle erwähnt, ändern sich Argumentationen im Laufe der Zeit und neue Aspekte werden immer wieder eingebracht. Eine hundertprozentige Klassifikation, ob etwas unterstützend oder angreifend ist, ist von den Daten und der Zeit abhängig. Einen idealen Korpus für Training und Klassifikation gibt es also in der Theorie nicht, da man auch Argumentationen nie eindeutig klassifizieren kann, um damit Relationen zu finden, denn diese sind auch abhängig vom Thema.

[1] `https://de.wikipedia.org/wiki/Tabakrauchen#Geschichte_des_Rauchens/`

6.2.3 Word2Vec

Word2Vec ist ein Framework, welches seit 2013 in vielen Forschungen mit Worteinbettung und neuronalen Netzen hoch angepriesen wird. Doch ist es wirklich das, was es verspricht? Oft wird auf das Paradebeispiel verwiesen mit den positiven Wortvektoren von 'woman' und 'king' mit Einbezug des negativen Wortvektor 'man', welches als Resultat 'queen' liefert. Dieses Ergebnis erscheint natürlich logisch und richtig. Will man aber nun andere logische Beispiele sich bilden, so stößt man oft auf schlechte Ergebnisse. Im folgenden wurden 6 Relationen gebildet:

```
A1) ['library'] <- ['book']
A2) ['library'] <- ['borrow','book']
B1) ['productivity'] <- ['employee']
B2) ['productivity'] <- ['hard','working','employee']
B3) ['productivity'] <- ['fool','around','employee']
B4) ['productivity'] <- ['lazy','employee']
```

Das Beispiel A1 soll zuerst zeigen, dass zwischen 'Buch' und 'Bibliothek' eine Relation besteht, diese aber durch den Begriff 'ausleihen' in Beispiel A2 verstärkt wird. Ähnliche Beispiele, in Bezug auf Arbeit und Produktivität sind von B1 bis B4 zu finden. B1 sollte einen neutralen Wert bilden, der durch B2 verstärkt wird, hingegen von B3 und B4 abgeschwächt werden sollte. In Tabelle 6.3 sind die Resultate für die einzelnen Beispiele und die berechnete Kosinusähnlichkeit auf den verschiedenen Modellen zu sehen.

	Trainingsdaten			
Relationen	uppsala	text8	wiki14	news
A1	0.2524	0.2548	0.4871	0.3245
A2	0.6919	0.2387	0.4871	0.3738
B1	0.1451	0.2264	0.4416	0.2474
B2	0.3316	0.2868	0.4623	0.2636
B3	0.1626	0.2218	0.3817	0.1666
B4	0.1472	0.2320	0.3772	0.3205

Tabelle 6.3: Kosinusähnlichkeitsresultate der Beispielrelationen A1 bis B4

Zu sehen ist, dass uppsala tatsächlich hier die besten Resultate liefert. Sowohl A2 als auch B2 haben wie erwartet höhere Werte, während aber B3 und B4 sich kaum schlechter zeigen als der neutrale Wert von B1. Der deutlich gute Wert bei A2, den auch nur uppsala erreicht hat, lässt sich wohl daraus schließen, dass es wirklich ein Essay gab im uppsala Trainingskorpus, welches

über Bücher, Bibliotheken und Ausleihe berichtete. Die anderen Modelle hatten zwar auch diese Wörter im Vokabular, aber entweder waren diese zu Allgemein, als das diese zu ausschlagenden Ergebnissen führen würden oder das Wort 'book' besitzt annähernd die selben Wortvektoren wie 'borrow', weshalb sich selbst bei der Verbindung dieser zweier Wortvektoren, kein großer Unterschied im Endergebnis herausstellt. In den B Beispielen lässt sich erkennen, dass in fast jedem Modell, der B2 Wert über den B3 und B4 Wert liegt, wie erwartet. Allerdings sind die Werte nur leicht besser und die Negativbeispiele B3 und B4 sind oftmals besser als der neutrale Wert B1, welches gegen die Erwartung ist. Die Ergebnisse sind also leicht durchwachsen und zeigen nur leichte Tendenzen zu den Erwartungen. Besonders schwierig ist es hier eine Grenzlinie zwischen 'positiven' und 'negativen' Beispielen zu ziehen, da diese Grenze vom Modell abhängig ist und oftmals nur sehr dünn vorliegt. Während in uppsala die Unterschiede von A1 zu A2 und B1 zu B2 gut erkennbar sind, so liegen die Werte in text8 alle recht nah beieinander. Aufbauend auf diese Erkenntnisse sollen im Folgenden alternative Lösungsansätze aufgezeigt werden.

Im letzten Absatz und den Experimentenresultaten aus dem Kapitel 5 'Experimente und Resultate' hat sich gezeigt, dass viele Ergebnisse oft nah beieinander liegen, obwohl diese nach erwarteter Relation sich stärker unterscheiden sollten. Eine Lösung für dieses Problem könnte das weiterführende Framework 'Doc2Vec' sein, erstmals beschrieben in 'Distributed Representations of Sentences and Documents' (Le and Mikolov [2014]). Obwohl dieses ähnlich arbeitet wie Word2Vec, werden hier für jeden Satz, Paragraphen oder Dokument (je nachdem wie man den Text vorher auftrennt) ein eigener Wortvektor gebildet. Dieser gehört nicht zu einem spezifischen Wort, sondern zum jeweiligen Abschnitt und wird auch nur dort trainiert. Dies hat den Vorteil, dass man nicht nur Vektoren für die Worte trainiert, sondern auch die Kontexte, in denen sie sich befinden. Durch die Abbildung 6.1 soll der Ablauf noch einmal verdeutlicht werden.

Dieses neue Modell kann Relationen aus verschiedenen Kontexten (am besten ein Kontext pro Dokument) lernen und damit auch besser klassifizieren. Leider konnte außer dem kleinen uppsala Trainingsset kein weiteres Modell mit Doc2Vec trainiert werden, da text8 als Rohdaten nicht in Abschnitten vorliegt und wiki14 zu groß war, um mit meinen möglichen technischen Mitteln, ein Modell daraus zu trainieren. Die Genauigkeiten mit dem Doc2Vec-uppsala-Modell haben sich nicht verbessert, was wohl daran liegt, dass die Hauptwörter der Essays nicht von verschiedenen Kontexten aufgegriffen werden.

Eine weitere Möglichkeit um Word2Vec in Bezug auf Relationen zu verbes-

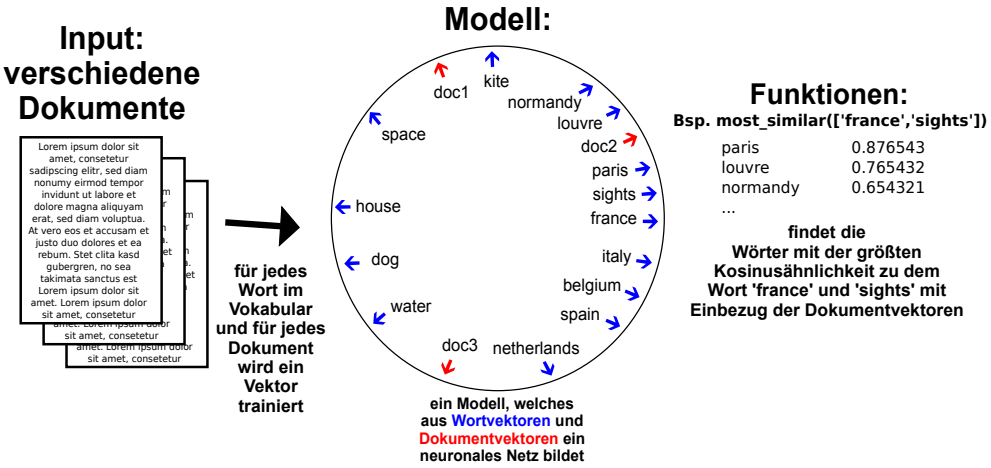

Abbildung 6.1: Ablauf von Doc2Vec

sern, ist durch das Prä-Training der Modelle mit 'Convolutional Neural Networks' (kurz CNN, Kim [2014]), welches von Yoon Kim 2014 entwickelt wurde. Durch dieses Framework soll für die 'universellen' Wortvektoren aus Word2Vec an bestimmten Klassifikationsaufgaben ein Feintuning vorgenommen werden. Ergebnisse mit Kosinusähnlichkeit zeigen, dass zum Beispiel bessere ähnliche Wörter zu 'good' und 'bad' gefunden wurden als vor dem Prä-Training mit CNN. In meiner Arbeit wurde das Prä-Training mit CNN nicht durchgeführt, weil die durch CNN modifizierten Modelle in einem eigenen Format vorlagen, die ich nicht für meine Experimente verwenden konnte.

Ein anderes Verfahren, um mit Word2Vec erstellte Wortvektoren zu verbessern, ist mit Hilfe von 'Retrofitting' (Faruqui et al. [2014]), welches Ende 2014 veröffentlicht wurde. In Gegensatz zu CNN werden hier die Wortvektoren nicht an spezifischen Klassifikationsaufgaben verfeinert, sondern Anhand eines semantischen Lexikons. Dieses beinhaltet alle Synonyme, Hyperonymien, Hyponymien und Umschreibungen von Wörtern. Das Ziel von Retrofitting ist, dass Wörter, die miteinander verbunden sind, auch (als Vektoren ausgedrückt) in die gleiche Richtung zeigen, zum Beispiel bei der Adjektiv- und Adverbform eines Wortes. Dies kann man auch als Weiterführung der Lemmatisierung (siehe Kapitel 3.1 'Datenextrahierung mit den Stanford CoreNLP Tools') ansehen, bei der nicht nur Mehrzahl und Fallanpassungen entfernt werden, sondern auch alle Arten von Synonymen mit den gleichen Wortvektoren versehen werden sollen. Obwohl dieses Framework ebenfalls sehr viel versprechend wirkt, wurde es

in meiner Arbeit auch nicht zur Anwendung gebracht. Gründe dafür sind auch hier der fehlerbehaftete Code, der einige Modelle aufgrund unterschiedlicher Formatierung nicht lesen bzw. verarbeiten konnte.

Kapitel 7

Zusammenfassung

In dieser Arbeit wurde versucht durch Worteinbettung als semantisches Feature die argumentativen Eigenschaften zwischen Segmenten oder ganzen Sätzen zu erkennen. Dabei wurden drei Experimente durchgeführt, die die automatische argumentative Analyse in drei Aspekten verbessern soll: die Textsegmentierung, die Segmentklassifikation und die Relationenfindung. Zur Evaluation der Experimente wurden zwei argumentative Korpora AAEC und AMC verwendet (siehe Kapitel 4.2). Dies hat sich als zu komplex herausgestellt und nur befriedigende Ergebnisse hervorgebracht, wodurch keine der ersten drei Thesen aufgezeigt werden konnte (siehe Kapitel 6.1). Vergleichsverfahren wie Bag-of-Words konnten dabei kaum geschlagen werden. Die verschiedenen Annahmen, die in These 4 bis 6 formuliert waren und sich mit Word2Vec beschäftigt haben, konnten sich dagegen bewahrheitet. Durch einen größeren Trainingsdatenkorpus und das Extrahieren von bestimmten Bestandteilen aus den Klassifikationsdaten, konnten bessere Ergebnisse erzielt werden. Auch konnte gezeigt werden, dass man ab einem bestimmten Grenzwert in der Kosinusähnlichkeit eine Relation als argumentativ einstufen kann oder nicht. Leider schwankt dieser Wert stark, je nach Modell und Aufgabe (siehe Kapitel 6.2.3).

Da Word2Vec selber nur schwer verschiedene Kontexte auseinander halten kann, sondern nur ein allgemeines Modell für alle Wörter aufstellt, ist es ratsam weiterführende Arbeiten zu verfolgen wie 'Paragraph2Vec' Le and Mikolov [2014] oder 'Retrofitting' Faruqui et al. [2014], die verschiedene Vektoren je nach Kontext für ein Wort aufstellen und Synonyme eines Wortes ebenfalls mit den gleichen Vektoren versehen. Weiterhin besteht aber das Problem, dass es keinen guten argumentativen Trainingskorpus für verschiedene Themen zum Trainieren gibt, bzw. es schwer ist neue Argumentationen tatsächlich vorherzusagen. Man müsste einen neuen Korpus erstellen, der sowohl die positiven Eigenschaften der hier verwendeten Trainingsdaten 'uppsala' und

'wiki14' enthält. Einerseits müsste er viele Themen umfassen wie wiki14, bei dem zu fast jeden Wort in den Wikipedia-Artikeln genügend Relationen gelernt werden können, andererseits dürfte er nur argumentative Texte beinhalten wie in uppsala, damit man auch wirklich die argumentativen Relationen zwischen den Wörtern lernt. Ein perfekter Korpus wäre also eine viel größere Variante von uppsala, in dem es mehr Essays und mehr Themen gibt, die auch ständig geupdatet werden.

Literaturverzeichnis

M. Azar. Argumentative text as rhetorical structure: An application of rhetorical structure theory. *Argumentation*, 13(1):97–114, 1999. doi: 10.1023/A:1007794409860. 2.1.3

Lucas Carstens and Francesca Toni. Towards relation based argumentation mining. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 29–34, Denver, CO, June 2015. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W15-0504`. 4.2, 4.2

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard H. Hovy, and Noah A. Smith. Retrofitting word vectors to semantic lexicons. *CoRR*, abs/1411.4166, 2014. URL `http://arxiv.org/abs/1411.4166`. 6.2.3, 7

Johannes Kiesel. Approaches to automatic argumentative unit segmentation, 2016. nicht veröffentlicht. 2.1.1, 4.2

Yoon Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014. URL `http://arxiv.org/abs/1408.5882`. 6.2.3

John Lawrence, Chris Reed, Colin Allen, Simon McAlister, and Andrew Ravenscroft. Mining arguments from 19th century philosophical texts using topic based modelling. In *Proceedings of the First Workshop on Argumentation Mining*, pages 79–87, Baltimore, Maryland, June 2014. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W/W14/W14-2111`. 2.1, 2.1.1

Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053, 2014. URL `http://arxiv.org/abs/1405.4053`. 3.2.2, 6.2.3, 7

William C. Mann and Sandra A. Thompson. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281, 1988. 2.1.3

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013. URL `http://arxiv.org/abs/1301.3781`. 1, 3.2

Huy Nguyen and Diane Litman. Extracting argument and domain words for identifying argument components in texts. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 22–28, Denver, CO, June 2015. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W15-0503`. 2.2

Christos Sardianos, Ioannis Manousos Katakis, Georgios Petasis, and Vangelis Karkaletsis. Argument extraction from news. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 56–66, Denver, CO, June 2015. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W15-0508`. 2.2

Adriaan M. J. Schakel and Benjamin J. Wilson. Measuring word significance using distributed representations of words. *CoRR*, abs/1508.02297, 2015. URL `http://arxiv.org/abs/1508.02297`. 3.2.2, 6.1.1

Parinaz Sobhani, Diana Inkpen, and Stan Matwin. From argumentation mining to stance classification. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 67–77, Denver, CO, June 2015. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W15-0509`. 2.2

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 926–934. Curran Associates, Inc., 2013. URL `http://papers.nips.cc/paper/5028-reasoning-with-neural-tensor-networks-for-knowledge-base-completion.pdf`. 6.2.2

Christian Stab and Iryna Gurevych. Annotating argument components and relations in persuasive essays. In Junichi Tsujii and Jan Hajic, editors, *Proceedings of the 25th International Conference on Computational Linguistics (COLING 2014)*, pages 1501–1510, Dublin, Ireland, August 2014. Dublin City University and Association for Computational Linguistics. 1, 2.1.3, 4.2, 4.2

Bonnie Webber, Markus Egg, and Valia Kordoni. Discourse structure and language technology. *Journal of Natural Language Engineering*, 01:1–40, 2012. 2.1.4